비공식 카카오톡 봇 만들기

by Dark Tornado

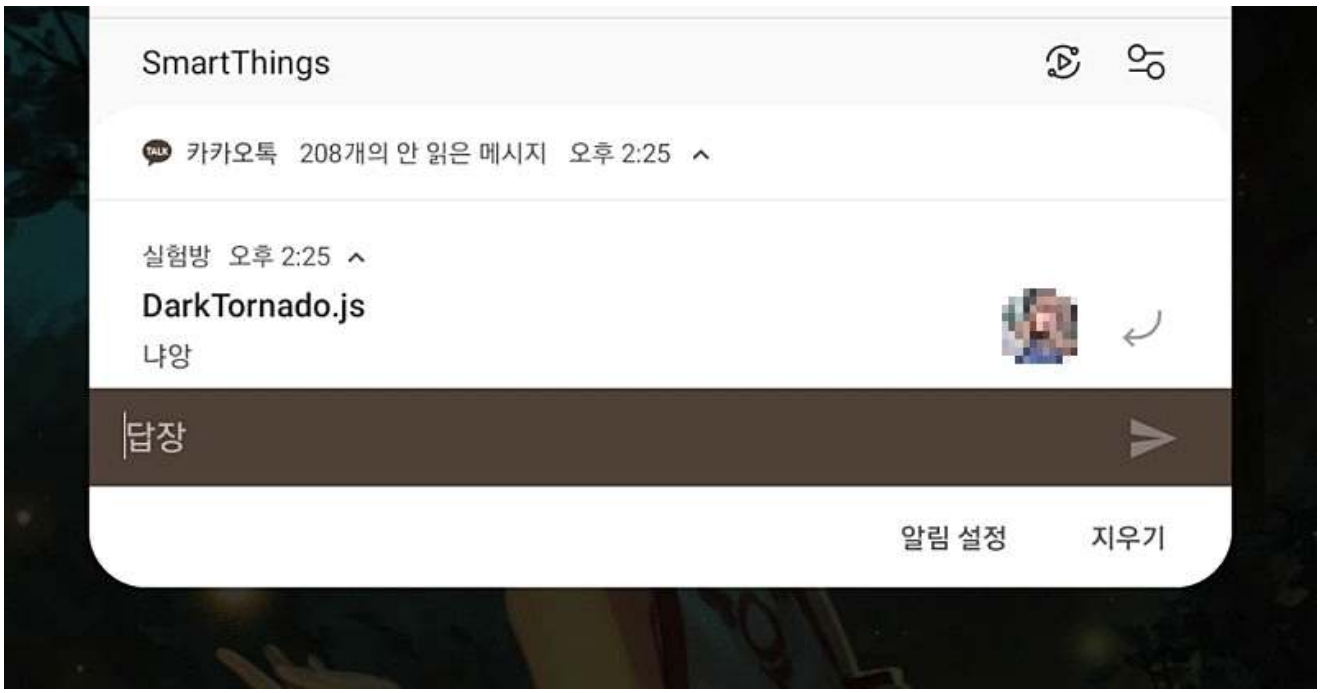
목차

서론

1. 봇 구동 방식과 구동 환경 구축

봇 구동 앱을 설치한다고 해서 채팅방에 봇 계정이 뿅 하고 나타나지는 않아요. 본인이 카카오톡을 보고 있지 않는 동안, 설정하거나 프로그래밍한 내용에 따라 봇 구동 앱이 자동으로 응답을 보내주는거에요.

채팅이 수신되면 상단바에 누가 어디서 뭐라고 보냈는지 알림이 뜨지요? 그리고, 상단바에서 바로 답장을 보내는 것이 가능하구요.



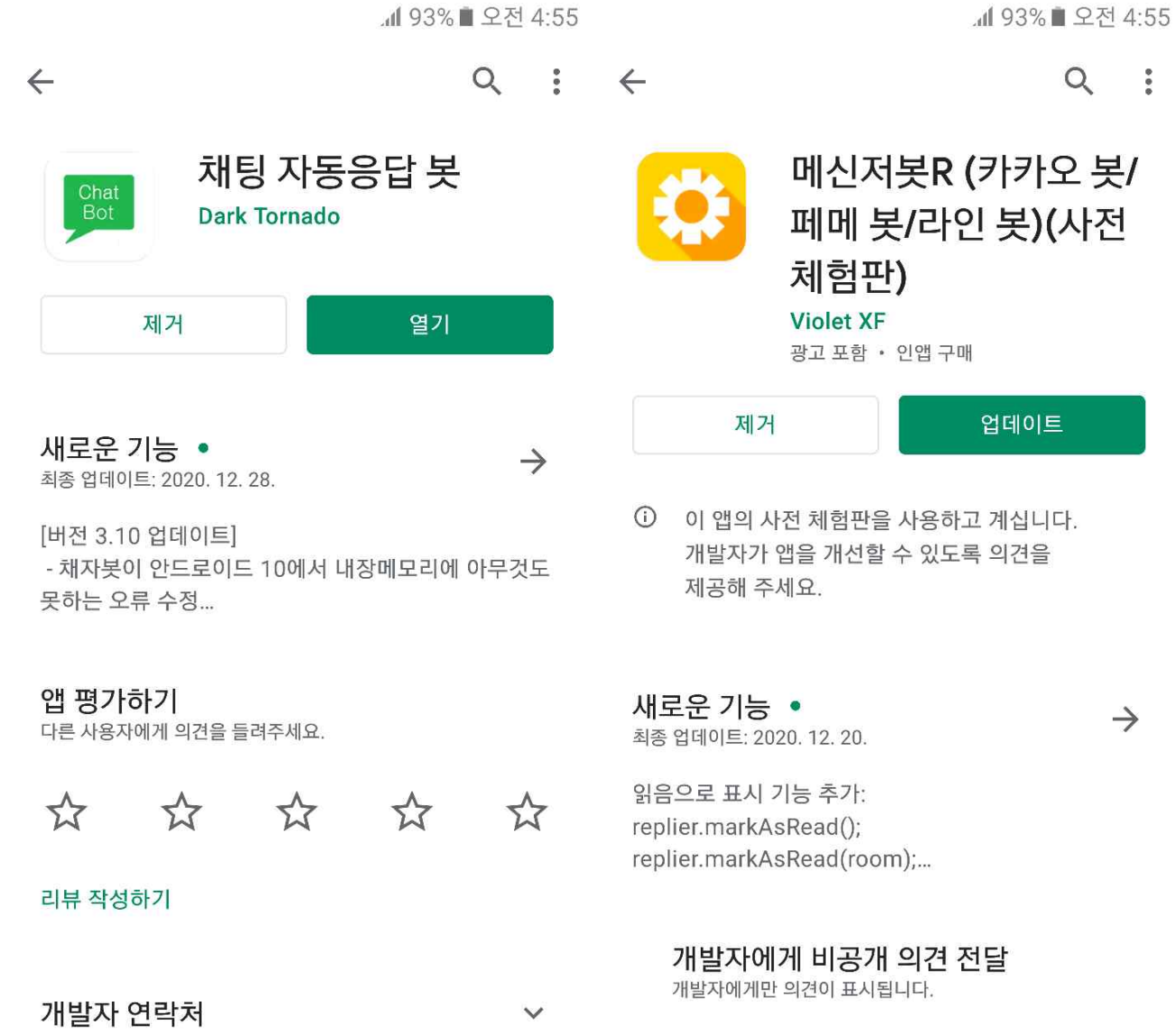
비공식 봇의 작동 방식도 위와 같아요. 채팅이 수신되어 상단바에 알림이 뜨면, 그 알림에 접근하여 내용을 분석, 누가 어디서 뭐라고 보냈는지 가지고 오는거에요. 그리고, 설정하거나 프로그래밍한 내용에 따라 해당 채팅방에 응답을 전송하거나, 전송하지 않는거에요.

그렇다면, 카카오톡 알림을 끈다면 봇이 작동하지 않겠지요? 알림은 뜨지만 내용이 뜨지 않도록 설정한 경우 역시 작동하지 않을 것이구요.

이 책에서는 자바스크립트를 통해 비공식 봇을 만드는 방법을 다루는거에요. 자바스크립트를 지원하는 비공식 봇 구동 앱들 중 많이 사용되는 것 앱은 '채팅 자동응답 봇'과 '메신저봇R'이에요.

이 책에서는 두 가지 봇 모두에서 사용할 수 있는 것들을 다룰 것이니, 무엇을 사용할지는 이것을 읽고 있는 여러분이 결정하시면 되는거에요. 두 가지 모두 Google Play 스토어에서 무료로

다운로드 받을 수 있어요.

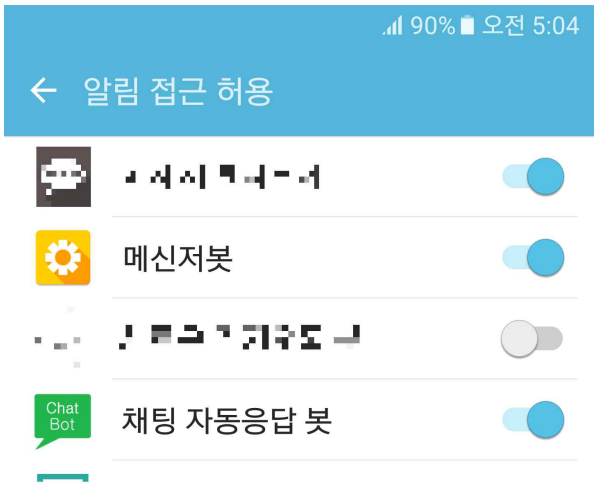


채팅 자동응답 봇

- 자바스크립트 말고도 커피스크립트, 루아, 비쥬얼 베이직을 지원해요.
- 단순 자동응답, 일반 자동응답 기능을 지원하기 때문에 프로그래밍 지식이 없어도 봇을 만들 수는 있어요.
- 다운로드 : https://play.google.com/store/apps/details?id=com.darktornado.chatbot

메신저봇R

- 자바스크립트만 지원하지만, 자바스크립트에 모든 것을 투자한 듯한 느낌인거에요.
- 따라서, 자바스크립트와 관련된 기능들은 다른 봇 구동 앱들 보다 많고 좋은거에요.
- 다운로드 : https://play.google.com/store/apps/details?id=com.xfl.msgbot



왼쪽 사진과 같이, 알림 접근 권한을 허용해주어야, 봇 구동 앱이 상단바에 뜬 카톡 알림에 접근할 수 있어요. 봇 구동 환경 구축 방법은 봇 구동 앱을 설치하시면 처음으로 뜨는 화면에서 알려주는거에요.

안드로이드 7 미만에서는, 지금은 Wear OS로 이름이 바뀐 안드로이드 웨어를 함께 설치해주어야 알림이 수신된 채팅방으로 채팅 전송이 가능해요. 설치하기만 하면 되는거에요.

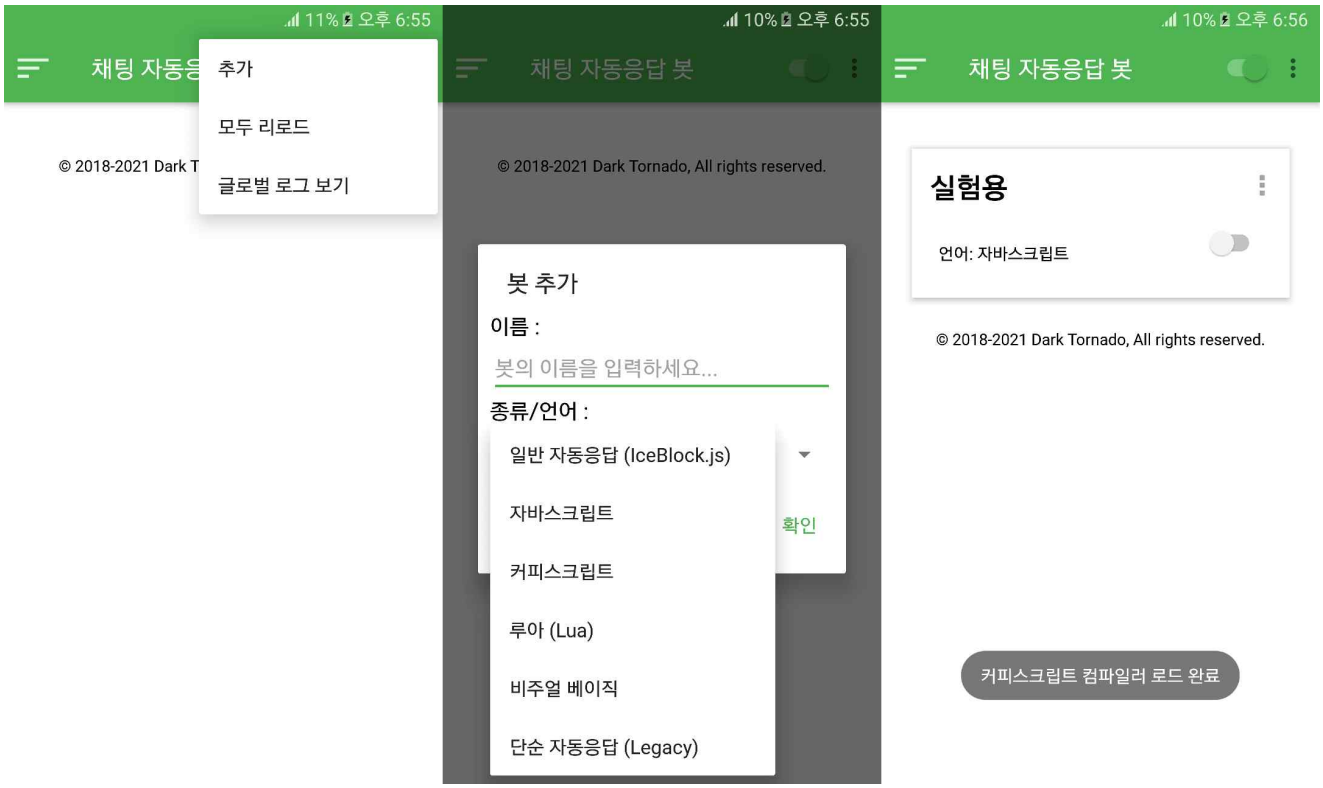
안드로이드 7 이상이여도 봇이 작동하지 않는 것 같다면 한 번 설지해보는 것도 좋고, 동일한 응답을 무조건 두 번씩 보낸다면 안드로이드 웨어가 원인일 수도 있으니 삭제하면 되는거에요.

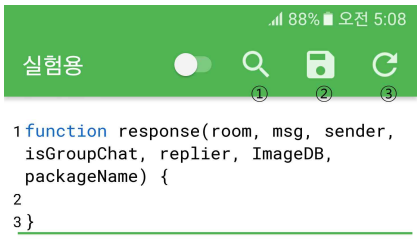
봇 추가하기

채팅 자동응답 봇에서는, 상단바 오른쪽에 있는 점 3개가 세로로 있는 버튼을 누르면 봇을 추가할 수 있어요. 이 책에서는 자바스크립트만 다룰 것이니, 자바스크립트를 선택하시면 되는거에요.

앱의 상단 부분에 있는 스위치가 봇 작동 여부를 결정하는 스위치이고, 봇마다 달려있는 스위치는 해당 봇의 작동 여부를 결정하는 스위치에요. 두 가지 모두 켜진 상태여야 봇이 작동해요.

추가된 이후에는 봇의 이름을 누르면 해당 봇의 편집 창으로 이동할 수 있어요.





엮 이미지는 자바스크립트를 사용하는 봇의 편집 화면 모습이에요. 봇의 소스를 수정하여 저장하거나, 리로드 등을 할 수 있어요.

이 화면에 있는 앱 상단의 스위치는 해당 봇의 작동 여부를 결정하는 스위치에요.

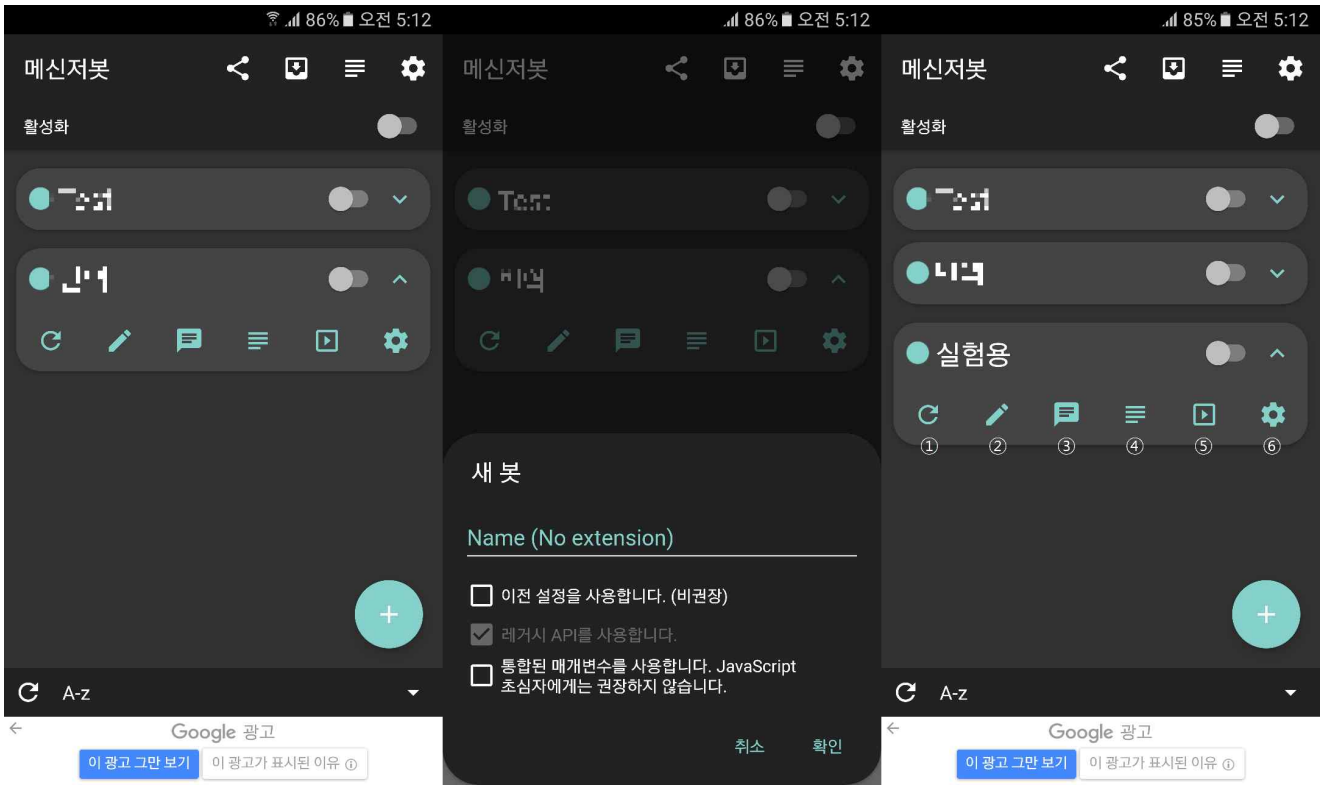
상단에 있는 다른 버튼들의 기능은 다음과 같아요.

① 찾기/바꾸기 등 메뉴
② 소스 저장
③ 리로드 (컴파일)

하단에 보이는 버튼들의 경우, 용도가 함께 적혀있으니 굳이 설명하지는 않을거에요.

엮 이미지에 보이는 소스는 봇을 생성하면 같이 생성되는 소스에요. 다른 소스를 복사 붙여넣기 할 때는 대부분 저 소스를 지우고 복사 붙여넣기 해야 해요.

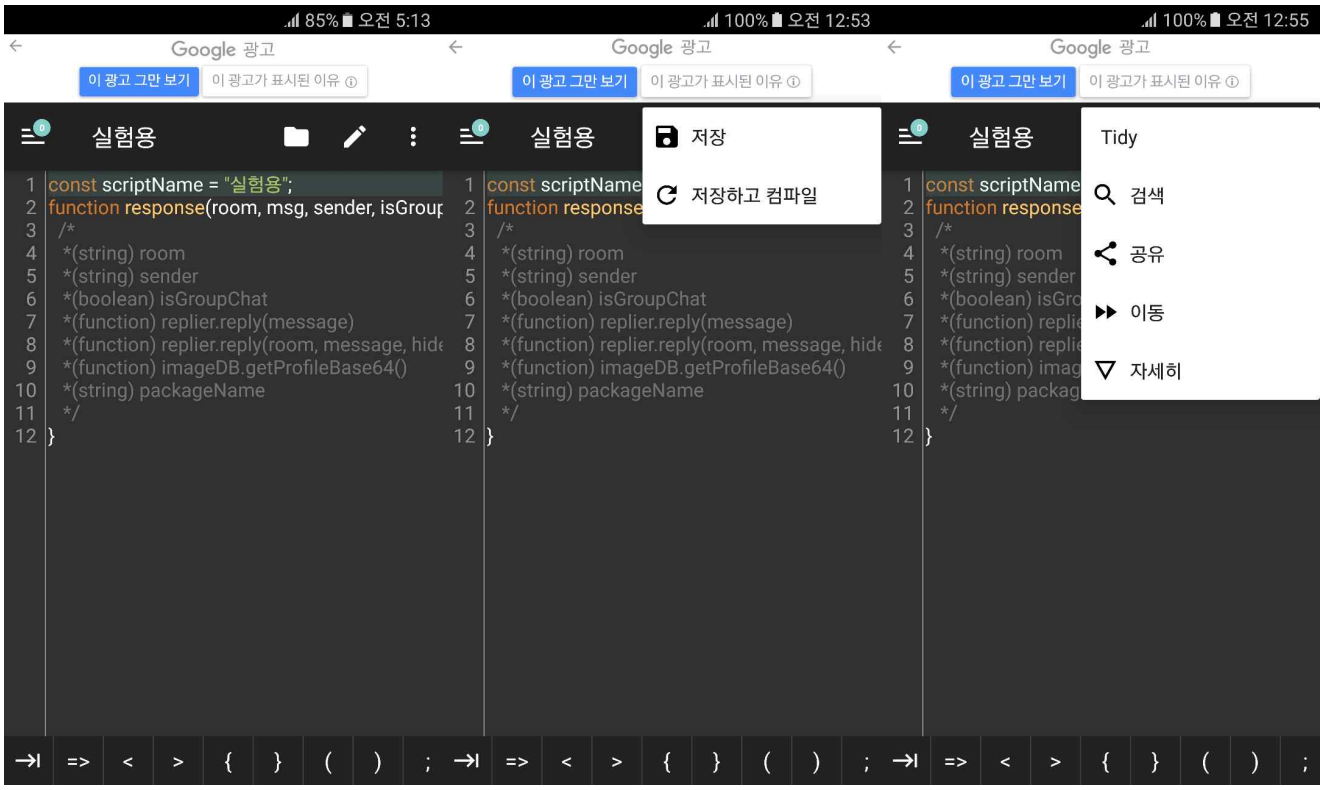
메신저봇R에서는 화면 오른쪽 아래의 + 버튼을 누르면 봇을 추가할 수 있어요. 앱의 상단 부분에 있는 스위치가 봇 작동 여부를 결정하는 스위치이고, 봇마다 달려있는 스위치는 해당 봇의 작동 여부를 결정하는 스위치에요. 두 가지 모두 켜진 상태여야 봇이 작동해요.



봇마다 있는 스위치의 오른쪽에 있는 버튼을 누르면 아래에 메뉴가 열리는거에요. 그 메뉴에 있는 버튼들의 용도는 다음과 같아요.

① 리로드 (컴파일)
② 소스 편집
③ 디버깅 룸
④ 로그 보기
⑤ 스크립트 액티비티 실행
⑥ 봇 설정

아래 이미지는 자바스크립트를 사용하는 봇의 편집 화면 모습이에요. 봇의 소스를 수정하여 저장하거나, 리로드 등을 할 수 있어요.
상단에 있는 다른 버튼들을 누르면 가운데, 오른쪽 이미지와 같이 기능 목록이 뜰거에요.



봇 구동 앱들은 자바스크립트 구동을 위해 대부분 라이노 자바스크립트 엔진을 사용하는거에요. 라이노 엔진 특성상 앱 내부의 클래스들에 접근할 수 있고, 자바 패키지를 포함한 안드로이드 앱 개발에 사용되는 패키지들을 자바스크립트 문법으로 사용할 수 있는거에요.
또한 채팅 자동응답 봇과 메신저봇R의 경우, 앱 내부에 자바로 만들어진 HTML 파싱 라이브러리인 jsoup이 내장되어 있기 때문에, 원래는 자바에서 사용하는 jsoup 라이브러리를 자바스크립트 문법으로 사용할 수 있어요.

2. 주석 등

개행 구분

자바스크립트는 ; 또는 엔터로 개행 구분을 해요. 어떠한 동작과 같은 것들을 구분한다고 생각하시면 쉬운거에요. ;를 생략한다고 해서 소스가 작동하지 않는 것은 아니지만, 붙이는 것을 권장하는거에요.

주석

주석은 기계가 읽을 때 무시하고 지나가는 부분이에요. 주로, 해당 영역에 대한 설명을 작성하거나, 소스 작성자가 누구인지와 같은 사람만 알면 되는 정보를 적을 때 사용되는거에요.

옆 이미지와 같이, //로 시작하는 경우, 해당 줄은 그 뒷부분부터 주석으로 처리되요.

/*와 */ 사이에 있는 영역 역시 주석으로 처리되구요.

가끔 일부 소스 하이라이터, 그러니끼 보기 편하게 소스에 있는 특정 단어들의 색상을 바꿔주거나 하는 기능이 "" 속에 있는 //도 주석으로 인식하는 경우도 있는데, 실제 실행될 때는 주석이 아닌 것으로 정상적으로 인식하는거에요.

리로드 (컴파일)

프로그래밍 언어로 작성된 소스 코드를 기계어로 번역하는 것을 컴파일이라고 불러요. 다른 프로그래밍 언어로 번역하거나, 기계어를 프로그래밍 언어로 번역하는 경우도 컴파일에 포함되지만, 일반적으로 컴파일이라고 하면 프로그래밍 언어로 작성된 소스를 기계어로 번역하는 것만 의미하는거에요.

다른 프로그래밍 언어로 번역하는 것은 트랜스컴파일, 기계어를 프로그래밍 언어로 번역하는 것은 디컴파일이라고 부르기도 하는거에요.

하지만, 그때그때 필요한 부분만 번역하여 실행하는 경우도 있어요. 일부분만 컴파일하고 나머지 부분은 그때그때 팔요한 번역하여 실행하는 경우도 있지요. 일단 반쯤 컴파일하고 그 반쯤 컴파일한걸 그때그때 필요한 부분만 번역하여 실행하는 경우도 있고, 그걸 또 일부분만 완전히 컴

파일해서 섞어서 사용하는 등, 경우가 많이 나뉘기에, 비공식 카카오톡 봇에서는 그냥 리로드라고 부를꺼에요.

자바스크립트

3. 변수와 상수, 자료형

변수

변수란, **어떠한 값을 저장하는 메모리 공간**을 의미해요. 메모리 공간에 저장된 값이 아니라, 그 메모리 1공간 자체를 의미해요.

프로그램을 실행하면 프로세스가 한 개 이상 실행되는데, 그 프로세스가 어떠한 값을 저장하기 위해서 운영체제에게 메모리 공간을 달라고 하면, 운영체제가 메모리 공간(RAM)의 일부분을 할당해주는거에요. 그 공간이 변수에요.

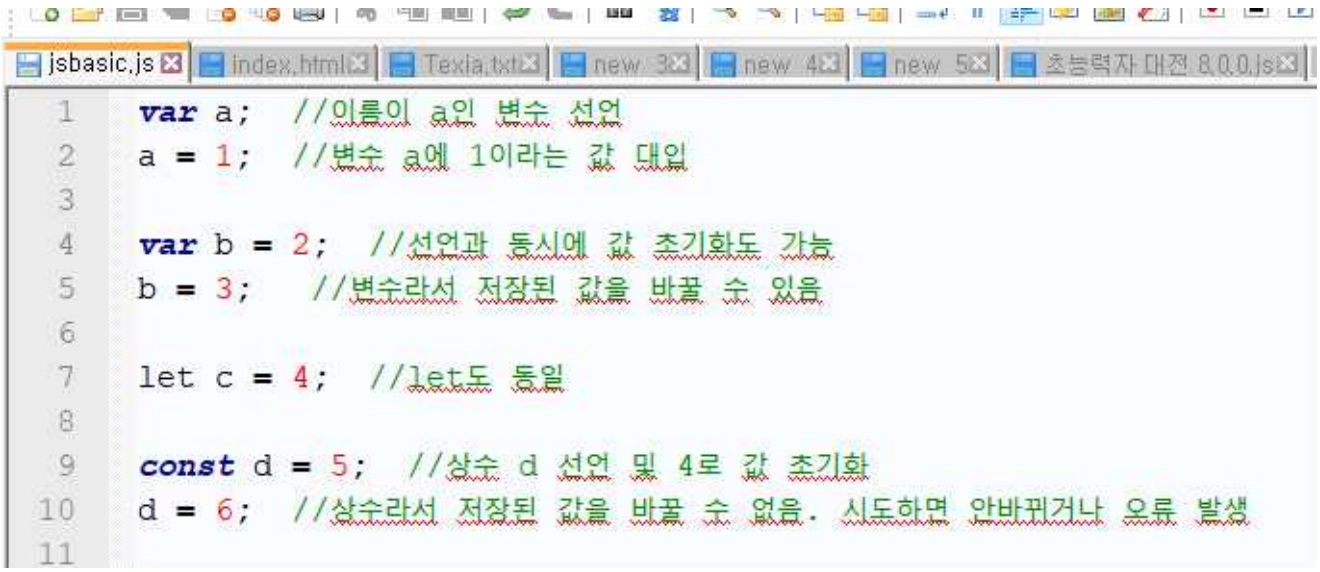
스코프

스코프란, **변수 등에 접근할 수 있는 범위**를 의미해요. 소스 코드 전체에서 존재하는 '전역 스코프'와, 일부 영역 내부(함수 또는 블록)에서만 존재하는 '지역 스코프'가 있어요.

전역 스코프에 있는 변수는 어디서나 접근할 수 있는 '전역 변수'가 되고, 지역 스코프에 있는 변수는 그 지연 속에서만 접근할 수 있는 '지역 변수'가 되요.

변수의 선언과 상수

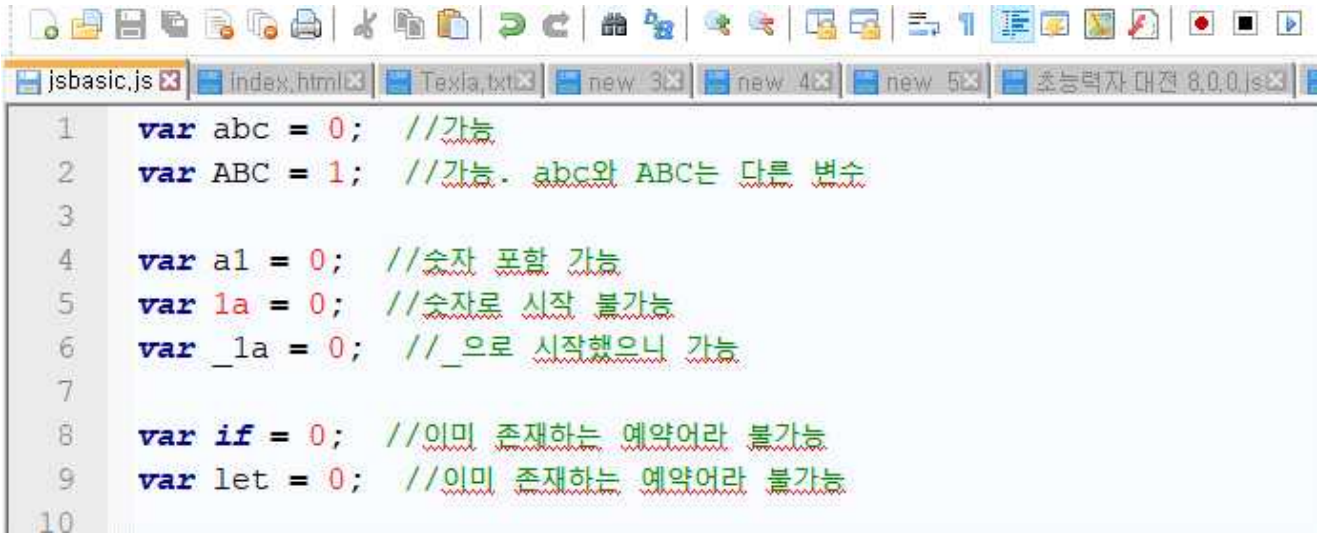
자바스크립트에서는 var 또는 let이라는 키워드로 변수를 선언할 수 있어요. const로 선언하는 경우, 값이 변하지 않는 상수가 되는거에요.



다음과 같이, 연속연산자인 ,를 이용하여 변수를 여러개 선언할 수도 있어요.

변수 이름 규칙

자바스크립트의 변수/상수의 이름은 대문자와 소문자를 구분해요. 변수의 이름에는 숫자가 들어갈 수 있지만, 숫자로 시작할 수는 없어요. 한글을 포함한 문자 또는 _, $로만 시작할 수 있는 거에요.

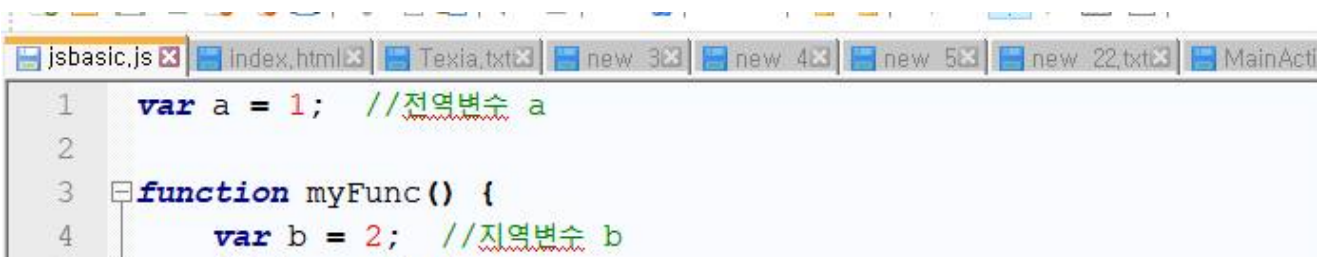


전역변수와 지역변수

전역에서 선언한 변수는 소스 코드 전체에서 존재하는 전역 변수가 되요. 전역이란, 어느 지역의 전체이자 전 지역의 함축된 단어로, 간단하게 중괄호로 묶어놓은 영역의 밖이라고 보면 되는 거에요.

지역에서 선언한 변수는 지역 변수가 되요. 지역변수는 그 지역을 벗어나게 되면 할당이 해제되어 사라지는거에요.

var로 선언한 변수는 함수 스코프를 가지기 때문에, 그 함수 안에서만 존재해요.



```
    if (a == 1) {
        var c = 3;  //지역변수 c
    }
    //function myFunc 안에서는 변수 a, b, c 모두 존재
}
//함수 밖 전역에서는 a만 존재
```

let으로 선언한 변수는 블록 스코프를 가지기 때문에, 그 블록 안에서만 존재해요. const 역시 let과 동일한 스코프를 가져요.

jsbasic.js | index.html | Texia.txt | new 3 | new 4 | new 5 | new 22.txt | MainActivity.java

```
let a = 1;  //전역변수 a

function myFunc() {
    let b = 2;  //지역변수 b
    if (a == 1) {
        let c = 3;  //지역변수 c
        //function myFunc 안에 있는 if 블록 안에서는 변수 a, b, c 모두 존재
    }
    //function myFunc 안에서는 변수 a, b만 존재
}
//모든 블록들의 밖 전역에서는 a만 존재
```

그리고, var과는 달리, let 또는 const로 선언한 경우, 동일한 이름을 가진 변수를 두 번 이상 선언하면 오류가 발생하는거에요.

jsbasic.js | index.html | Texia.txt | new 3 | new 4 | new 5

```
var a = 0;
var a = 1;  //오류 X

let b = 0;
let b = 1;  //오류 O
```

호이스팅과 var를 생략한 경우

자바스크립트에는 호이스팅이라는, **변수 또는 함수를 선언하는 부분을 위로 끌어올리는 것**이 존재해요. 원래는 var로 선언한 변수만 호이스팅 대상이지만, 봇 구동 앱에서 자바스크립트 구동을 위해 사용하는 라이노 자바스크립트 엔진의 경우. let이나 const 선언한 변수/상수까지 호이스

팅하는거에요.

지역에서도 var를 생략하면서 변수를 선언하면 전역변수가 된다는 주장이 있지만, 사실은 그렇게 보일 뿐, 전역변수가 되는 것이 아니에요. **var를 생략하면 변수가 선언되는 것이 아니라, 전역 객체에 속성으로 추가**되는거에요.

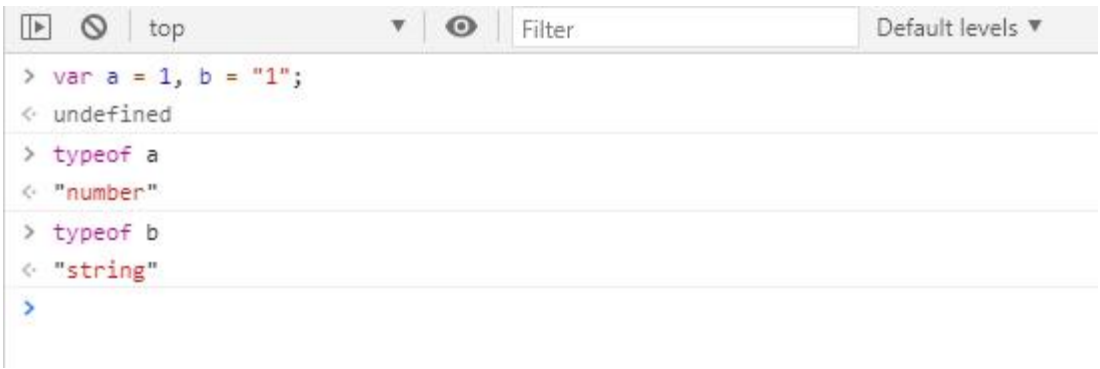
var나 let, const를 생략하는 문법은 자바스크립트에 존재하지 않으며, 단순한 값 대입인거에요. 호이스팅도 되지 않아요. 전역 객체에 속성으로 추가된 것이기에, 전역 객체(웹은 window, 라이노는 global)에 접근해보면 추가된 것을 확인할 수 있고, 객체의 속성이기에 delete로 지워지는거에요.

자료형

자료형(Data Type)이란, 논리값, 숫자, 문자열, null과 같은 **값의 종류**를 의미해요.

논리값(boolean)	참과 거짓. true와 false밖에 없어요.
숫자(number)	말 그대로 숫자. 1, 2, 3 같은거에요.
문자열(string)	" 또는 '로 시작하고 닫히는 문자열이에요.
null	값이 없음을 명시적으로 나타내는거에요.
undefined	진짜로 값이 없는거에요. 값을 할당하지 않으면 undefined가 할당되는거에요
객체	객체. 나중에 따로 설명할거에요.

다음과 같이, typeof를 통해 해당 값의 자료형을 알 수 있어요



형변환

형변환(Casting)이란, **다른 자료형으로 번경하는 것**을 의미하는거에요. 예를 들어, 수신된 채팅의 내용이 숫자로만 이루어져 있어도 자료형은 문자열일 수도 있는데, 이것을 수로 바꾸어서 연산할 때 등에 사용되는거에요.

논리값으로 변환	Boolean(값);
수로 변환	Number(값); 0을 빼거나 앞에 +를 붙이는 방법도 존재
문자열로 변환	String(값); 또는 값.toString(); 빈 문자열을 붙이는 법도 존재

라이노 엔진을 사용하는 비공식 카카오톡 봇 특정상 자바에만 존재하는 자료형인 float나 long으로 형변환을 해야하는 경우도 있는데, 이 경우는 자바에 존재하는 자동 박싱/언박싱을 이용하면 되는거에요.

이 책에서는 굳이 다루지 않을거에요.

4. 배열과 객체

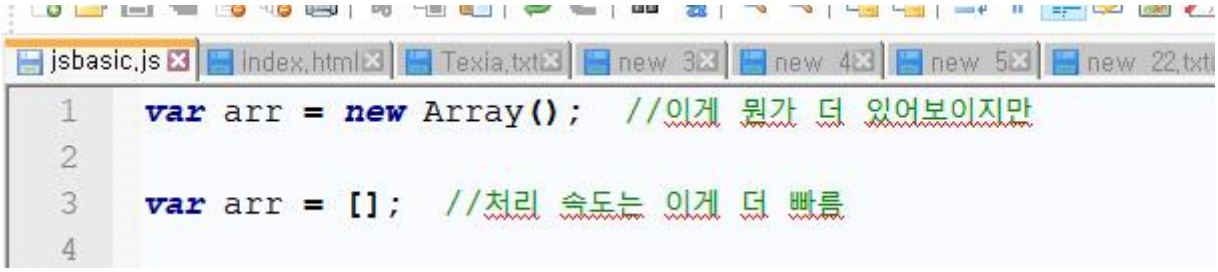
배열이란?

이름 하나에 값을 하나만 넣을 수 있는 변수와는 달리, **배열은 이름 하나에 값 여러 개를 넣을 수 있어요**. 넣은 값들은 순서(인덱스)로 구분하는거에요. 배열 arr의 0번째 값, 1번째 값, 2번째 값...과 같이 구분하고, 자바스크립트를 포함한 대부분의 프로그래밍 언어들에서 인덱스는 0부터 시작해요.

반복문 중 for문과 함께 쓰이는 경우가 많아요.

배열의 선언

빈 배열을 선언하는 방법은 총 두 가지가 있는데, new Array() 보다 []로 선언하는 것이 처리 속도가 더 빠른거에요.



new Array()로 선언할 때, 인자로 정수 하나만 넘기면 해당 길이 만큼인 배열이 만들어지고, 정수를 두 개 이상 또는 수가 아닌 값을 넘기거나 []로 선언하는 경우, 배열에 저장될 내용물도 함께 선언할 수 있어요.

```
1  var arr = new Array(3);  //길이가 3인 배열
2
3  var arr = new Array(1, 3, 5);  //0번째 값은 1, 1번째 값은 3, 2번째 값은 5
4  var arr = [1, 3, 5];  //위와 동일
5
```

배열 다루기

자바스크립트에서의 배열은 C나 C++, 자바에 있는 배열과는 달리 길이 제한이 없어요. 앞이나 뒤, 중간에 값을 추가하거나 삭제하는 것이 가능한거에요.

배열의 길이는 .length 속성을 통해 가지고 올 수 있어요. 배열이름.length와 같은 방식으로 사용하면 되는거에요.

jsbasic.js | index.html | Texia.txt | new 3 | new 4 | new 5 | new 22.txt | MainActivity.

```
var arr = [1, 2, 3];
arr[0] = 4; //배열의 0번째 요소에 4 대입.(순서는 0부터 시작)
결과 : [4, 2, 3]

var arr = [1, 2, 3];
arr.push(4); //배열의 가장 뒤에 4 추가
결과 : [1, 2, 3, 4]

var arr = [1, 2, 3];
arr.unshift(4); //배열의 가장 앞에 4 추가
결과 : [4, 1, 2, 3]

var arr = [1, 2, 3];
arr.pop(); //배열의 가장 뒤에 있는 요소 삭제
결과 : [1, 2]

var arr = [1, 2, 3];
arr.shift(4); //배열의 가장 앞에 있는 요소 삭제
결과 : [2, 3]

var arr = [1, 2, 3, 4, 5];
arr.splice(1, 2); //배열의 1번째 요소부터 2개 요소 삭제
결과 : [1, 4, 5]

var arr = [1, 2, 3, 4, 5];
arr.splice(1, 0, 6); //배열의 1번째에 6 추가
.splice();에서 0개 지우면 됨
결과 : [1, 6, 2, 3, 4, 5]

var arr = [1, 2, 3, 4, 5];
arr.indexOf(3); //3이라는 값이 어디에 있는지 가져옴
값이 없으면 -1 반환
결과 : 2
```

객체란?

배열에서는 저장된 값에 접근하기 위해 정수(인덱스)를 사용했었지요? **인덱스가 들어갈 자리에 정수 대신에 문자열을 넣어서 사용**할 수도 있는데, 이것을 객체라고 불러요.

변수에 변수를 넣은 것이라고 볼 수도 있는거에요. 함수도 넣을 수 있어요.

객체 사용해보기

```
1  var obj = {}; //빈 객체 선언
2
3  abj["a"] = 1; //객체에 a라는 이름을 저장된 값에 접근
4  obj.a = 1;    //이런식으로도 가능
5
```

이런식으로 선언과 동시에 내용물을 넣을 수도 있어요. 물론, 선언할 때 미리 다 넣었다고 해서 선언이 끝난 뒤에 넣거나 바꿀 수 없는건 아니에요.

```
1  var car = { //자동차 객체
2      speed: 0, //속도
3      driver: "누군가", //운전자
4      passenger: ["이늠", "저늠"], //탑승자들
5  };
```

이런식으로 변수 말고도 나중에 다루게 될 함수도 넣을 수 있어요.

```
1  var obj = {}; //빈 객체 선언
2  abj.value = "변수";
3  obj.func = function() {
4      //함수
5  };
```

배열의 비밀

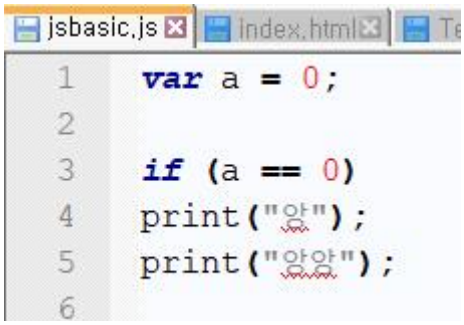
사실, 자바스크립트에 존재하는 배열은 배열처럼 작동하는 객체에요. 객체에서 가능한 것들은 배열에서도 가능한거에요.

5. 제어문과 연산자

제어문은 **소스 코드의 흐름을 제어**하는 것들이에요.

조건문 - if

조건을 확인한 뒤, 그 조건이 참이라면 다음에 오는 동작을 실행해요.



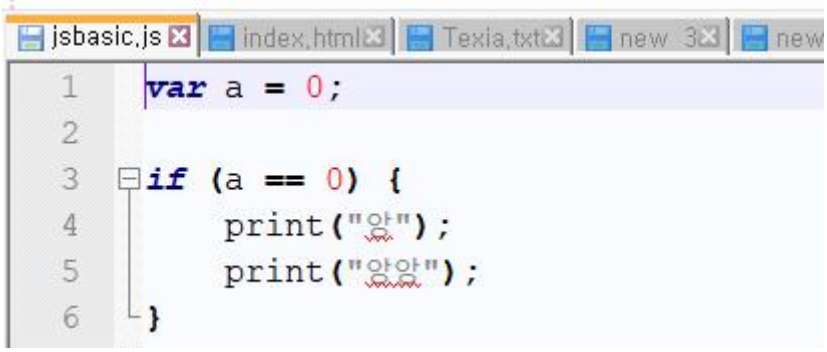
옆 이미지에 있는 소스를 예시로 보면, 1번째 줄에서 변수 a를 선언함과 동시에 0으로 초기화시켜요.

3번째 줄 if문은 변수 a에 저장된 값이 0인지 확인하는 부분이에요. 실제로 a에 저장된 값은 0이니 참이 될 것이고, 그 다음에 오는 동작인 print("앙"); 부분이 실행될거에요.

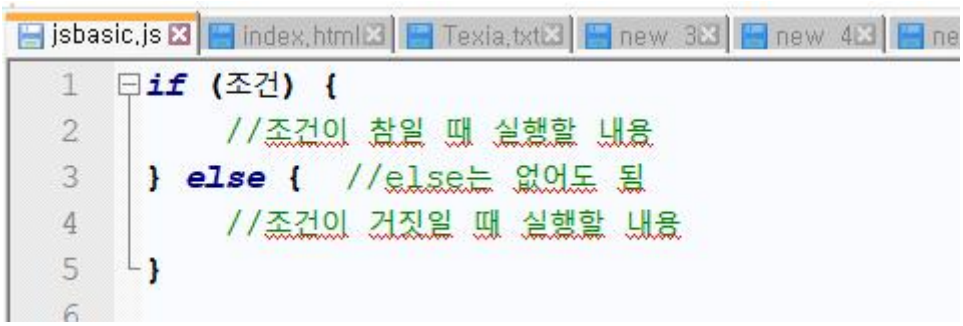
5번째 줄은 if문 바로 직후에 온 동작이 아니기 때문에, 조건이 거짓이여도 작동할거에요.

print();는 예시로 넣은거에요. 메신저봇R에는 print();가 존재하지 않아요

여러 동작들을 넣고 싶다면, 다음과 같이 중괄호로 묶으면 되는거에요.



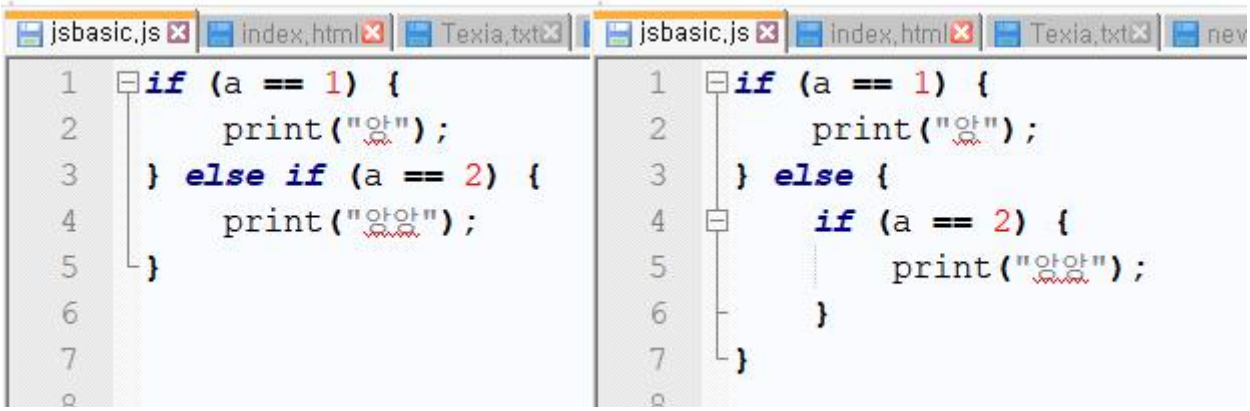
else와 함께 쓰이기도 해요. 이게 기본적인 틀은 다음과 같아요



조건 부분에 굳이 조건이 들어갈 필요는 없어요. 그냥 값을 넣어도 되는데, 그러면 그 값이 참

으로 취급되는 값인지, 거짓으로 취급되는지에 따라 조건이 참 또는 거짓으로 결정되는거에요.

else if라는 것도 존재하는 것처럼 보이지만, 실제로는 else 안에 if가 들어간거에요. 즉, 왼쪽 소스는 사실 오른쪽 소스처럼 되어있는거에요.



연산자

연산자는 **연산을 할 때 사용**되는거에요.

연산자	설명	연산자	설명
=	좌변에 우변 대입	==	좌변과 우변이 같다
+	더하기 또는 붙이기	===	좌변과 우변이 완전히 같다
-	빼기	!=	좌변과 우변이 다르다
*	곱하기	!==	좌변과 우변이 조금이라도 다르다
/	나누기	>	좌변이 우변 초과
%	나머지	>=	좌변이 우변 이상
++	변수에 저장된 값 1 증가	<	좌변이 우변 미만
--	변수에 저장된 값 1 감소	<=	좌변이 우변 이하
+=	좌변에 우변을 더하거나 붙인 값 저장	&&	그리고
-=	좌변에서 우변을 뺀 값 저장	||	또는
*=	좌변에 우변을 곱한 값 저장	,	연속연산자
/=	좌변에 우변을 나눈 값 저장	? :	삼항연산자

비트 연산자도 있고, typeof나 delete, instanceof도 연산자에 포함되지만, 주로 사용되는 것들만 표에 넣어두었어요.

조건문 - switch

switch는 **주로 메뉴를 구성할 때 사용**하는거에요. 오른쪽 소스에 있는 switch는 왼쪽 소스에 있는 if와 거의 동일하게 작동해요.

jsbasic.js | index.html | Texla.txt | new 3

```
if (a == 1) {
    print("앙");
} else if (a == 2) {
    print("앙앙");
} else {
    print("앙앙앙");
}
```

jsbasic.js | index.html | Texla.txt | new 3 | new

```
switch (a) {
    case 1:
        print("앙");
        break;
    case 2:
        print("앙앙");
        break;
    default:
        print("앙앙앙");
}
```

반복문 - while

특정한 조건을 만족하면 특정한 동작을 실행하는 것을 반복해요.

if와 비슷하지만, 조건을 만족하면 한 번 실행하고 끝나는 if와는 달리, while은 조건을 만족하면 실행하고, 다시 조건을 확인해서 만족하면 실행하고... 조건이 거짓이 될 때 까지 반복해서 실행하는거에요. 따라서, 조건 부분에 대놓고 true라고 적어버리면 무한 반복이 되지요.

```
var a = 0;
while (a < 5) {
    //무언가
    a++;
}

while (true) {
    //무한반복
}

var a = 0;
while (a < 5) {
    if (a == 2) break;
    //무언가
    a++;
}
```

왼쪽 소스 중 위에 있는 while의 경우, 변수 a의 값이 5 미만일 때 조건을 만족하고, 반복할 때 마다 변수 a의 값이 1씩 증가하기 때문에, 결과적으로는 무언가를 5번 실행하고 종료되는 거에요. 여기서 변수 a를 **카운터**라고 불러요.

가운데에 있는 while의 경우, 조건에 true라고 적어버렸으니 조건이 거짓이 될 일이 없기 때문에 무한 반복이 되는거에요.

아래에 있는 while의 경우, a가 2이면 반복문을 탈출하도록 하는 break;가 실행되기에, 무언가는 두 번만 실행되는거에요.

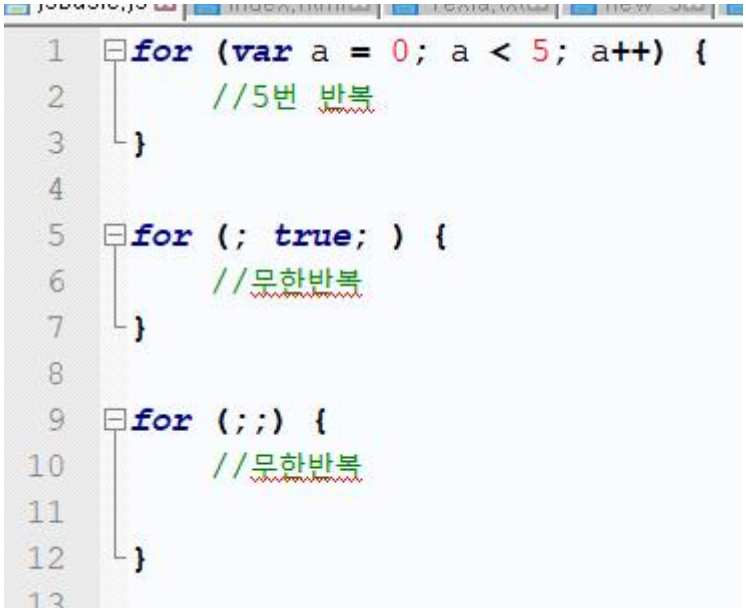
break는 위에서 다룬 것처럼 switch를 끝낼 때도 사용되고, 곧 다룰 for를 끝낼 때도 사용되는거에요.

도중에 반복을 멈추는 break 말고도, 해당

반복 회차를 건너뛰는 continue라는 것도 있는거에요.

반복문 - for

따로 **카운터를 만들어서 몇 번 반복해야 하는 경우가 많았기에, 그것을 모두 합친 for가 등장**한거에요. 기본적인 틀은 다음과 같아요. 초기화, 조건, 증감 부분 모두 생략할 수 있어요.



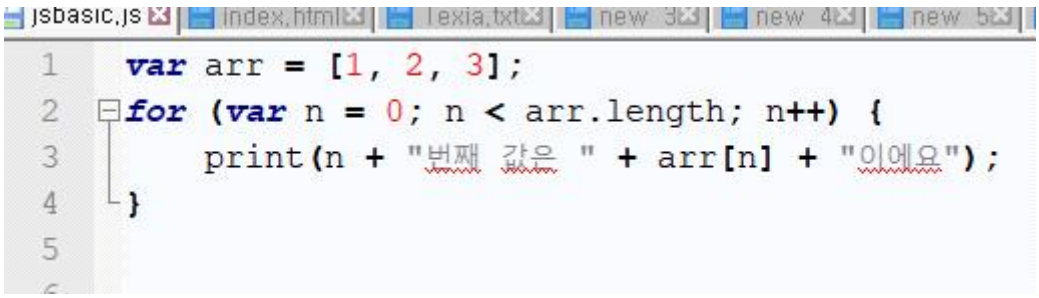
왼쪽 소스 중 위에 있는 for의 경우, 변수 위에서 다룬 while처럼 5번 반복하는거에요. 조건과 카운터 변수의 선언 및 초기화, 증감을 한 번에 적은거에요.

가운데에 있는 for의 경우, 조건 부분에 true라고 적어버렸기에 무한반복이 되는거에요.

아래에 있는 for의 경우, 조건 부분을 생략하면 참인 것으로 취급되기에, 이 역시 무한반복이 되는거에요.

그리고, 다음과 같이 배열에 있는 모든 값에 접근할 때 응용할 수 있는거에요. 배열의 길이보다 작은지를 조건 부분에 넣고, 배열의 인덱스 부분에 카운터를 넣으면 배열의 길이만큼 반복하면서 배열에 있는 모든 값에 접근할 수 있어요.

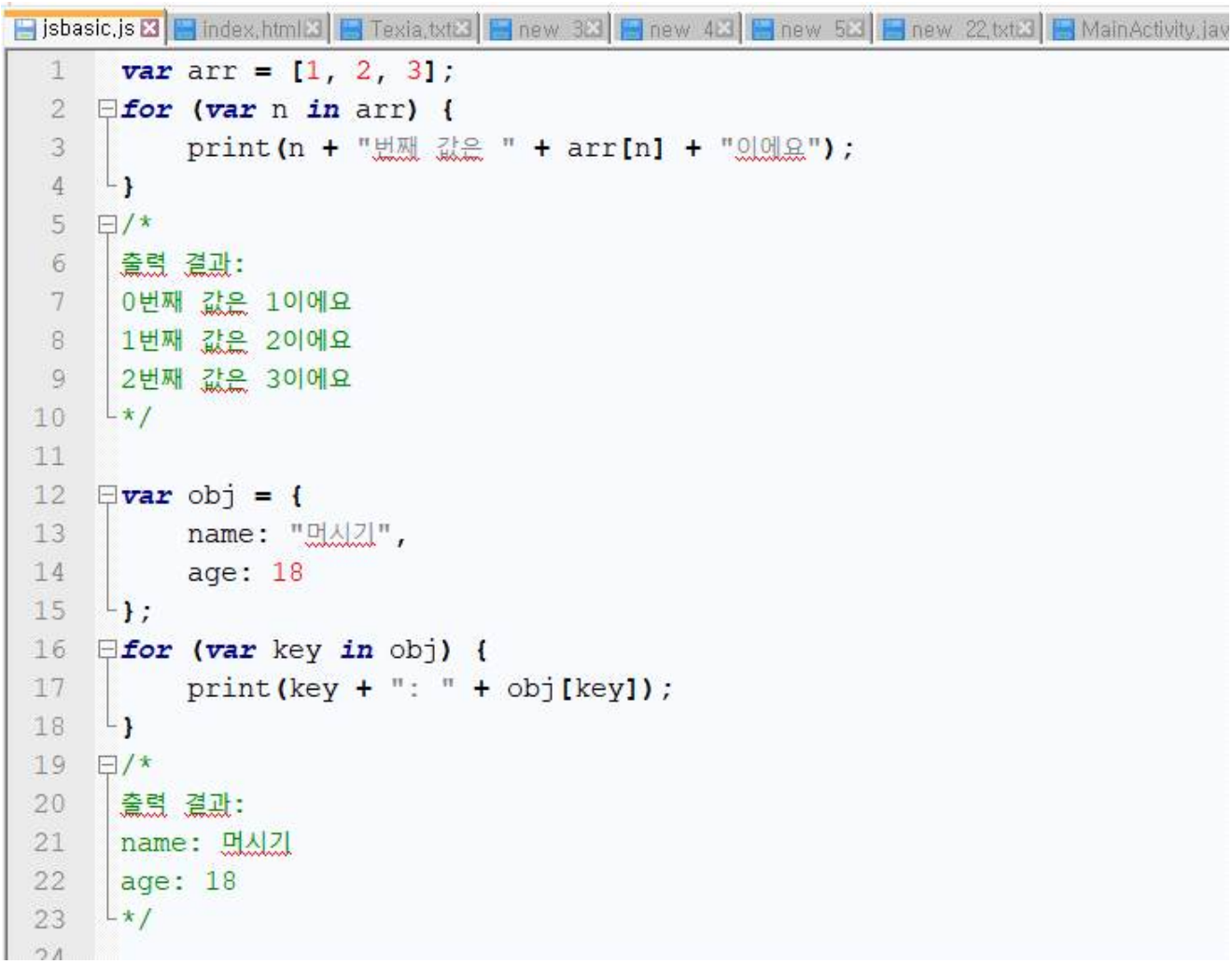
배열 안에 배열이 들어간 2차원 배열의 경우는 for 안에 for를 넣은 2중 for문을 사용하면 되구요.



for ... in

배열의 모든 값들에 접근하면서 반복하는 것처럼, **객체의 모든 값에 접근하여 반복**하는 for ... in도 있는거에요.

배열도 결국은 객체 기반인지라 배열에도 사용할 수 있지만, 카운터 변수의 자료형이 수가 아니라 문자열이에요.



6. 함수

함수란?

함수는 **일정한 또는 특정한 기능을 실행하는 소스 코드 블록**이에요. 예를 들어, 어떠한 문자열을 넘기면 그 문자열을 채팅방에 전송하는 동작(기능)을 하는 것이 존재하지요.

함수 정의해보기

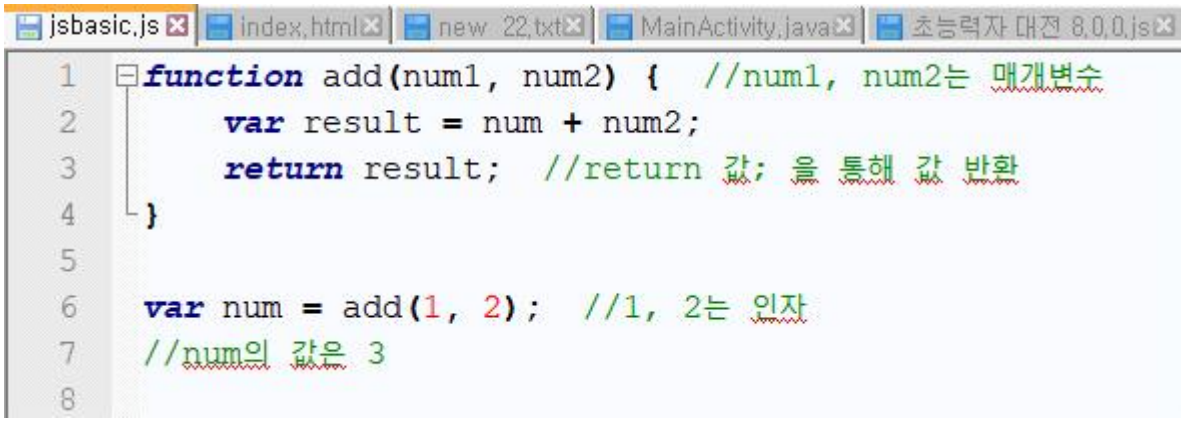
변수를 var라는 것을 사용하여 선언한 것처럼 함수는 **function을 사용해서 선언(정의)**할 수 있어요. 이름이 없는 함수를 변수에 대입하는 방법으로도 선언할 수 있어요.

함수 호출해보기

다음과 같이, **함수의 이름을 통해 함수를 호출**할 수 있어요.

매개변수와 인자

함수를 호출할 때 값을 넘겨야 하는 경우도 있는데, 이때 매개변수/인자를 사용하는거에요. **값을 넘길 때는 인자이고, 받을 때는 매개변수**에요. 매개변수들의 이름은 변수처럼 마음대로 지어도 되는거에요. 매개변수들의 값은 이름이 아니라 위치가 기준이에요.



매개변수는 지역변수처럼 함수 스코프를 가지고, 함수 내부에서 값을 바꿀 수도 있어요.

카카오톡 봇

7. 이벤트 리스너

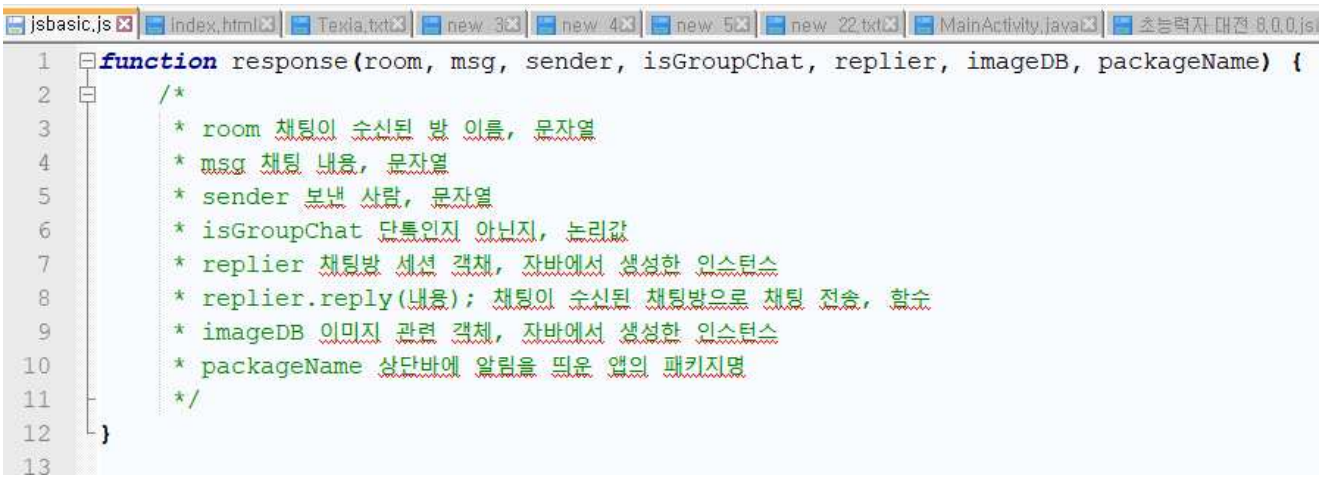
이벤트 리스너란?

특정한 이벤트가 발생하면 호출되는 함수에요. 함수처럼 생겼고, 함수처럼 호출할 수도 있어요.

정확히는, 봇 구동 앱의 경우 안드로이드(자바)쪽에서 자바스크립트 쪽 함수를 호출하면서 인자로 값들을 넘기는거에요. 가장 많이 사용되고 채팅 자동응답 봇과 메신저봇R 모두에 존재하는 이벤트 리스터 2가지를 설명할거에요.

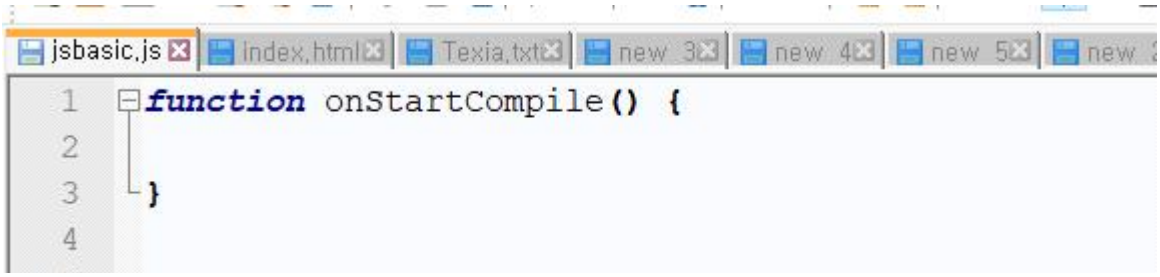
response

response는 **채팅이 수신되면**, 정확히는 카카오톡 또는 미리 지정해둔 메신저 앱이 상단바에 알림을 띄우면 **호출되는 이벤트 리스너**에요. 가장 많이 사용될거에요.



onStartCompile

onStartCompile는 **리로드 직전에 호출되는 이벤트 리스너**에요.

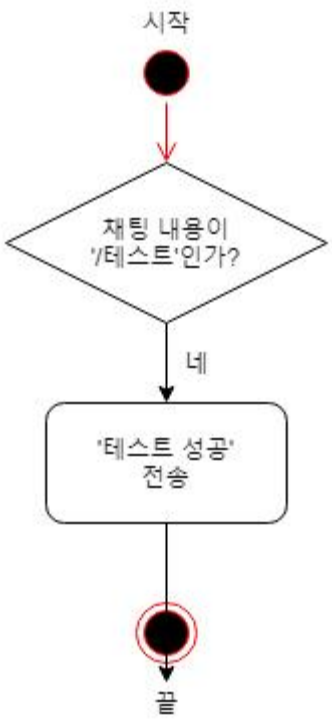


8. 특정 채팅이 수신되면 반응

간단하게, 특정 채팅이 수신되면 응답을 보내는 것을 해볼거에요. 누군가가 "/테스트"라는 채팅을 보내면 "테스트 성공"이라고 채팅을 보낼거에요.

1. 작동 방식 구상

어떻게 작동할지는 그림을 그리면서 생각해보면 편한거에요. 나중에는 그림을 그리지 않고도 할 수 있고, 언제부턴가 바로 소스 코드로 떠오르는 순간도 올거에요.



2. 필요한 것들

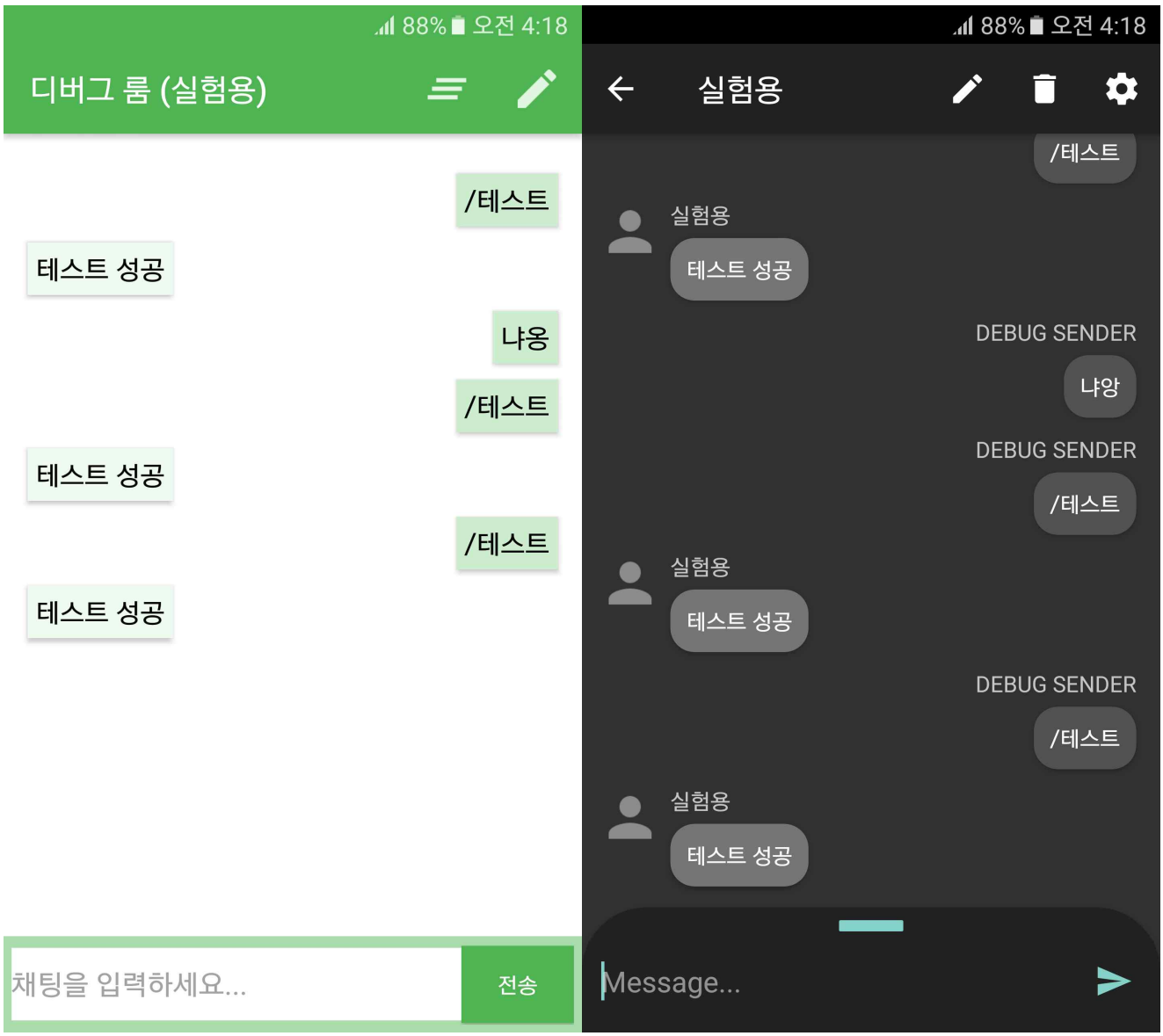
채팅이 수신되면 작동하도록 할 것이니, 이벤트 리스너 response가 필요할 것이고, 2번째 매개변수가 수신된 채팅 내용이니, 연산자 ==를 이용하여 값을 비교하는 것을 if 안에 넣고, 그 if의 영향권 안에 채팅 전송 부분을 넣으면 되겠지요?

3. 구현

```
jsbasic.js | index.html | Texia.txt | new 3 | new 4 | new 5 | new 22.txt | MainActivity.ja
1  function response(room, msg, sender, isGroupChat, replier) {
2      if (msg == "/테스트") { //수신된 채팅 내용이 '/테스트'라면
3          replier.reply("테스트 성공"); //'테스트 성공'이라고 답변
4      }
5  }
6
```

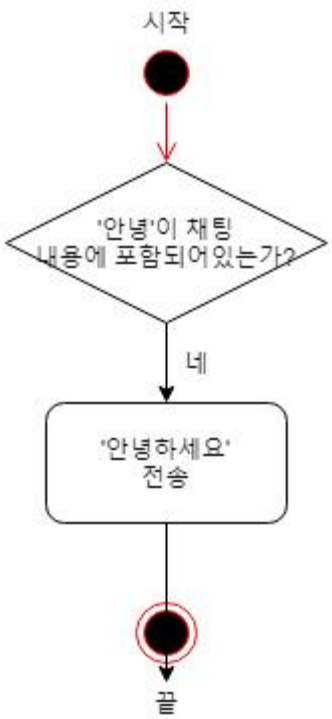
4. 실행 모습

잘 작동하는거에요. 봇 구동 앱의 디버깅 룸을 사용하면 카카오톡 없이도 봇 작동 여부를 확인할 수 있어요.



9. 특정 채팅이 포함되면 반응

1. 작동 방식 구상



2. 필요한 것들

저번과 동일하지만, if의 조건 부분에 들어갈 부분만 바꾸면 될 것 같은거에요. 배열에 있는 .length 속성을 사용하면 배열의 길이를 가지고 올 수 있었던 것처럼, 문자열에 있는 .includes(); 메소드를 사용하면 해당 문자열에 특정 문자열이 포함되어 있는지 확인할 수 있어요.

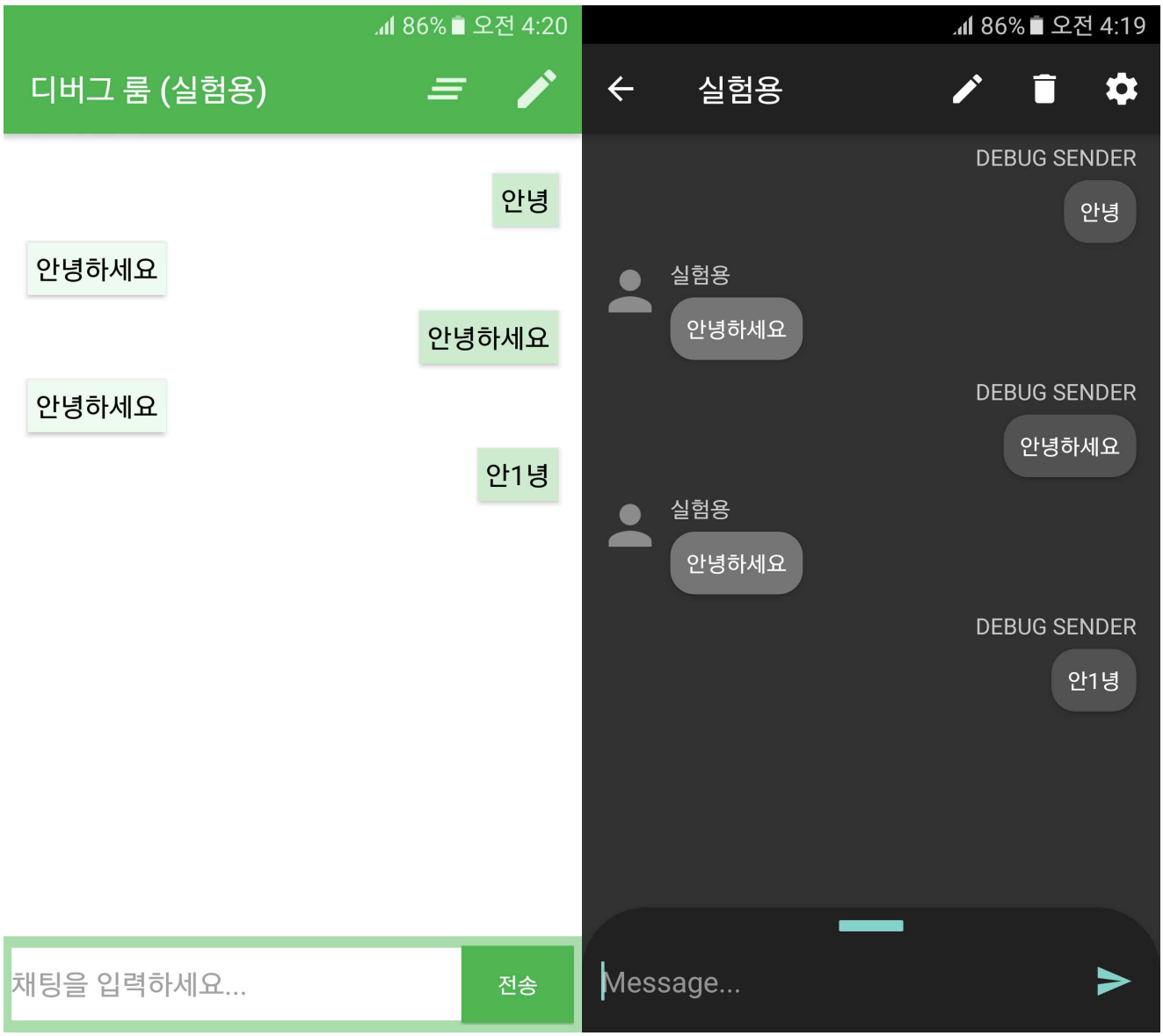
구상하다가 막히는 부분은 네2버나 9글에 검색하면 대부분 나오는거에요. 예를 들어 지금 같은 경우는 '자바스크립트 문자열 포함' 같은걸 검색하면 되는거에요.

3. 구현

jsbasic.js | index.html | Texia.txt | new 3 | new 4 | new 5 | new 22.txt | MainActivity.java

```
1  function response(room, msg, sender, isGroupChat, replier) {
2      if (msg.includes("안녕")) { //수신된 채팅에 '안녕'이 포함되어 있다면
3          replier.reply("안녕하세요"); //'안녕하세요'라고 답변
4      }
5  }
6
```

4. 실행 모습



위 사진들과 같이 잘 작동하는거에요. 하지만, 이러한 봇은 채팅방에 둘 이상 있다면 서로의 '안녕'에 반응하여 무한 반복을 하면서 계속 안녕하세요라는 채팅을 보낼거에요.

안녕봇(인사봇)이나 욕감지 봇, 일상적인 문구에 반응하는 봇들은 자신이 들어가 있는 채팅방에 민폐를 끼치기에 만들지 않는 것이 좋은거에요.

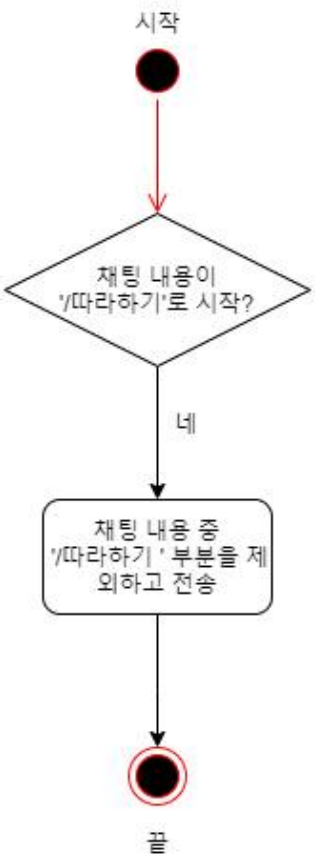
응용

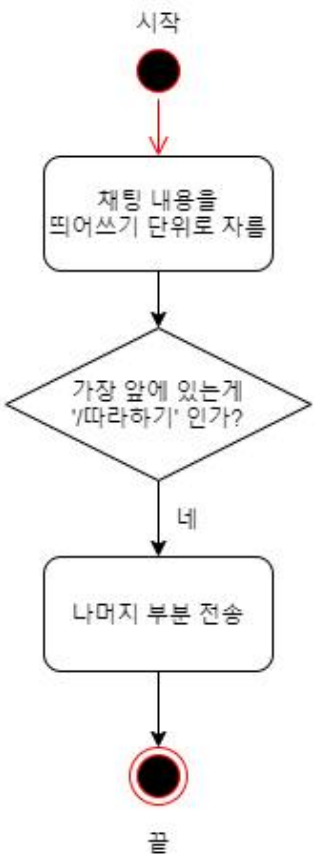
10. 따라하기

1. 작동 방식 구상

"/따라하기 123"이라고 말하면 봇이 "123"이라고 말하고, "/따라하기 앙"이라고 말하면 봇이 "앙"이라고 말하는 기능을 만들어볼거에요. 정확히는 수신된 채팅 내용이 "/따라하기 "로 시작하면 그 뒷부분을 전송하는 기능이지요.

방법은 크게 두 가지로 나뉘는거에요. 시작 부분이 일치하는지 확인하거나, 띄어쓰기 단위로 잘라서 비교하거나. 아무튼 두 가지 다 구현해볼거에요.





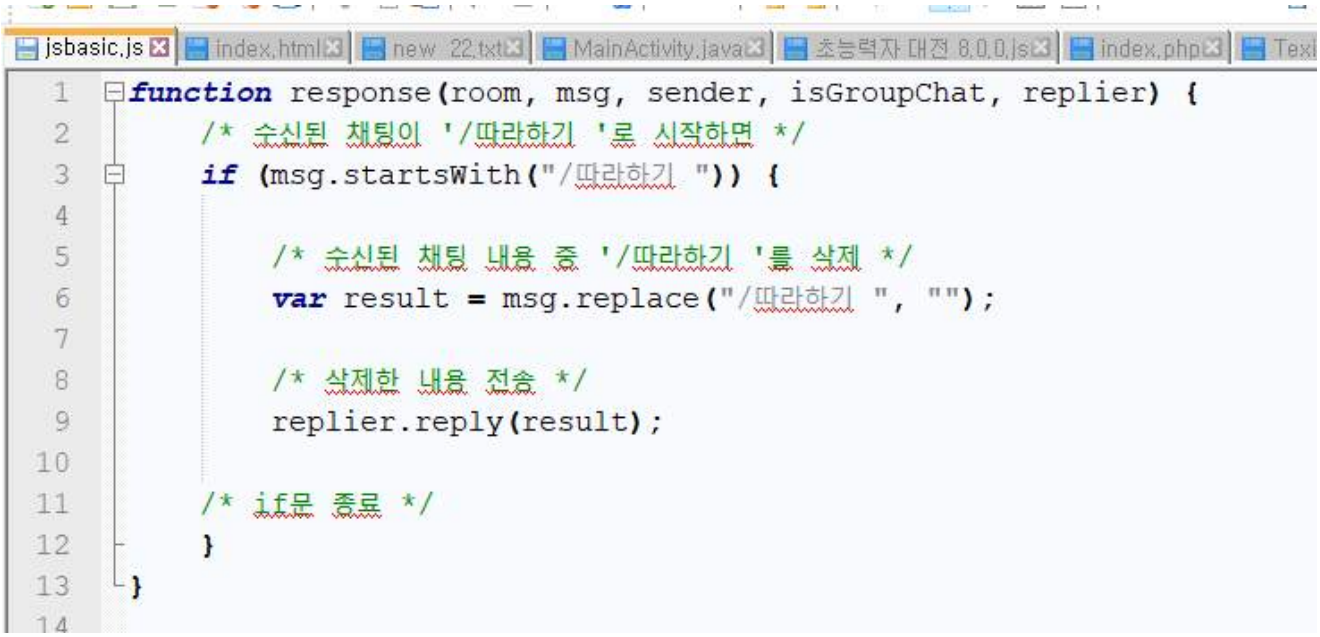
2. 필요한 것들

문자열에 있는 .startsWith(); 메소드를 사용하면 시작 부분이 일치하는지 확인할 수 있고, .replace(); 메소드를 사용하면 문자열에 있는 특정 문구를 다른 문구로 바꿀 수 있어요. 그 다른 문구가 빈 문자열이라면 삭제되는거지요.

그리고, .split(); 메소드를 사용하면 특정한 문구를 기준으로 해당 문자열을 잘라서 배열로 만들 수 있어요.

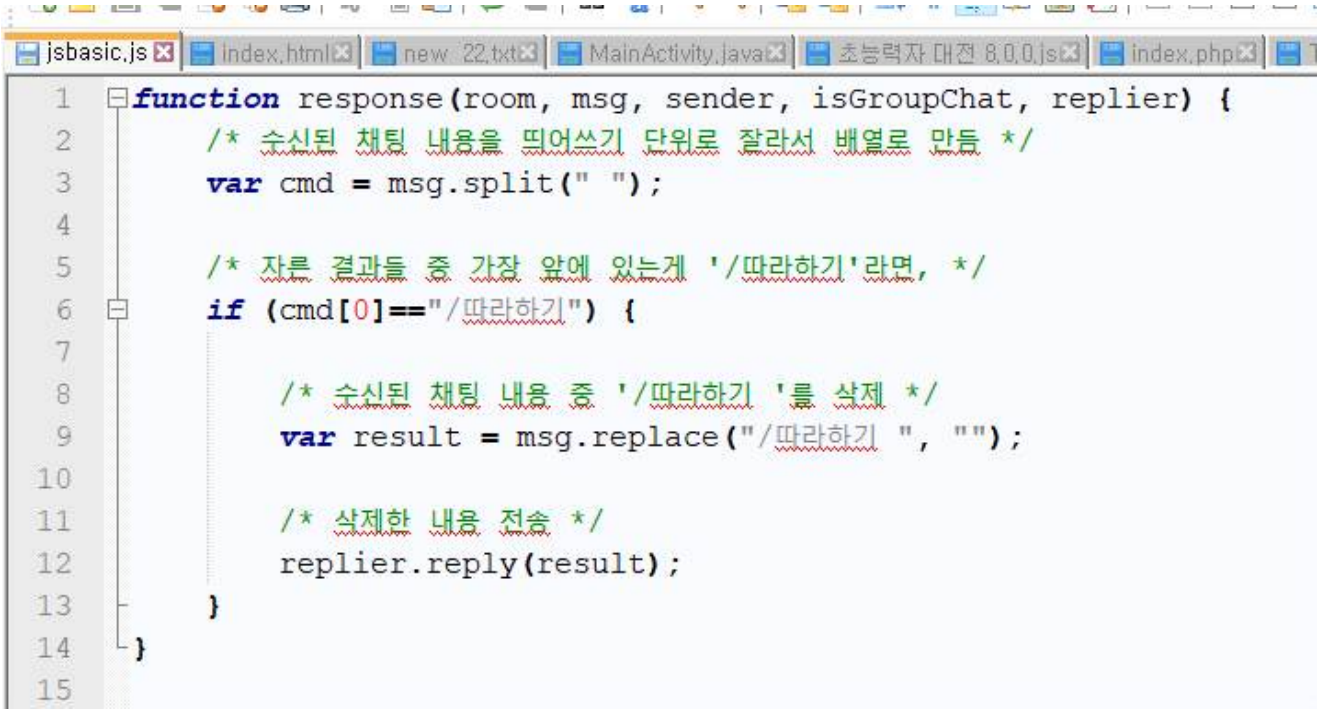
3. 구현

.startsWith(); 사용



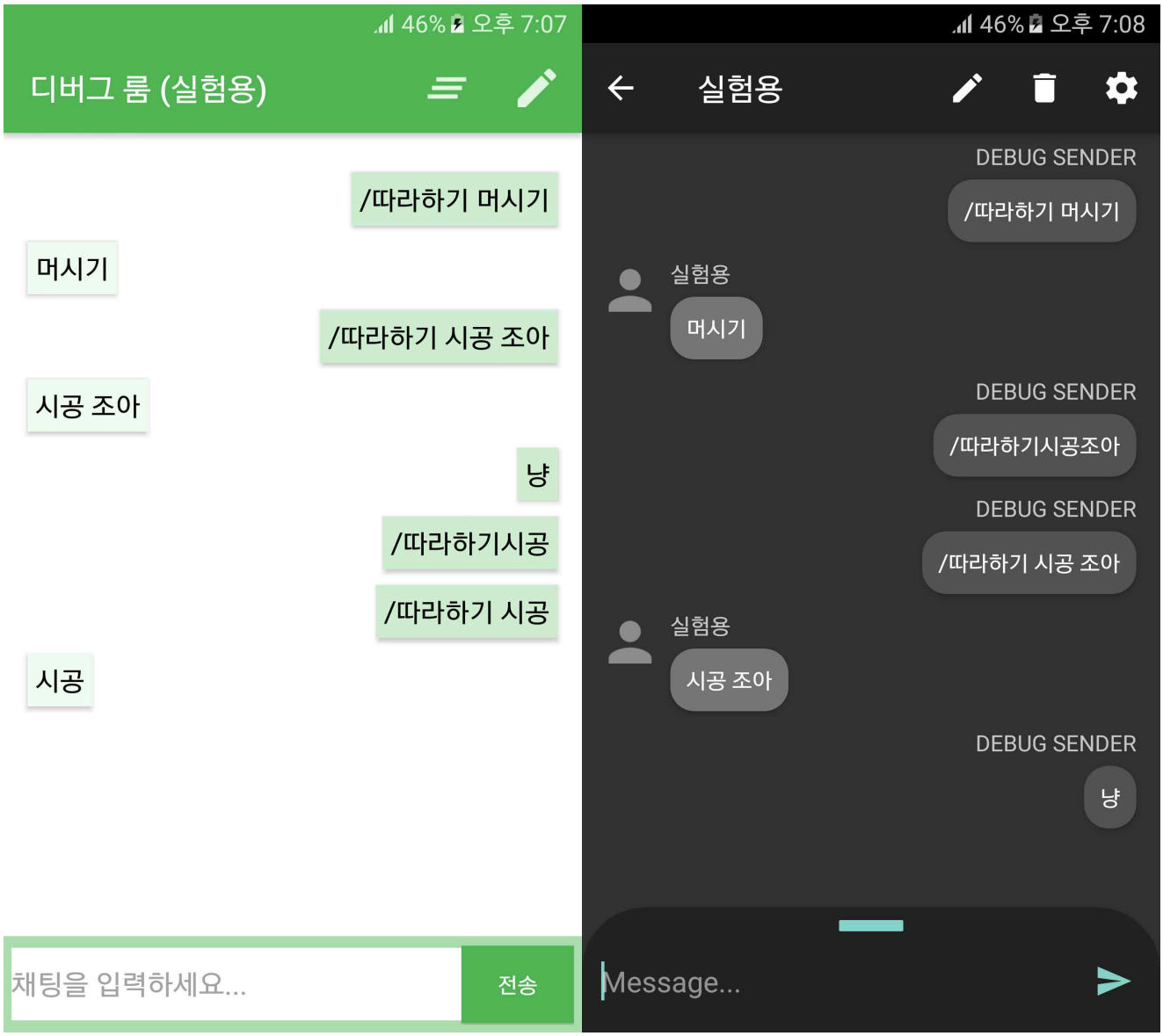
```
function response(room, msg, sender, isGroupChat, replier) {
    /* 수신된 채팅이 '/따라하기 '로 시작하면 */
    if (msg.startsWith("/따라하기 ")) {

        /* 수신된 채팅 내용 중 '/따라하기 '를 삭제 */
        var result = msg.replace("/따라하기 ", "");

        /* 삭제한 내용 전송 */
        replier.reply(result);

    /* if문 종료 */
    }
}
```

.split(); 사용



```
function response(room, msg, sender, isGroupChat, replier) {
    /* 수신된 채팅 내용을 띄어쓰기 단위로 잘라서 배열로 만듦 */
    var cmd = msg.split(" ");

    /* 자른 결과들 중 가장 앞에 있는게 '/따라하기'라면, */
    if (cmd[0]=="/따라하기") {

        /* 수신된 채팅 내용 중 '/따라하기 '를 삭제 */
        var result = msg.replace("/따라하기 ", "");

        /* 삭제한 내용 전송 */
        replier.reply(result);
    }
}
```

4. 실행 모습


46% 오후 7:07
디버그 룸 (실험용)
/따라하기 머시기
머시기
/따라하기 시공 조아
시공 조아
냥
/따라하기시공
/따라하기 시공
시공
채팅을 입력하세요...
전송
46% 오후 7:08
실험용
DEBUG SENDER
/따라하기 머시기
실험용
머시기
DEBUG SENDER
/따라하기시공조아
DEBUG SENDER
/따라하기 시공 조아
실험용
시공 조아
DEBUG SENDER
냥
Message...

11. 번역

1. 작동 방식 구상

이제는 굳이 그림을 그리지 않을거에요. "/영한 [영어]"을 보내면 해당 영어를 한국어로 번역한 결과를 보내고, "/한영 [한국어]"를 보내면 해당 한국어를 영어로 번역하도록 만들거에요.

2. 필요한 것들

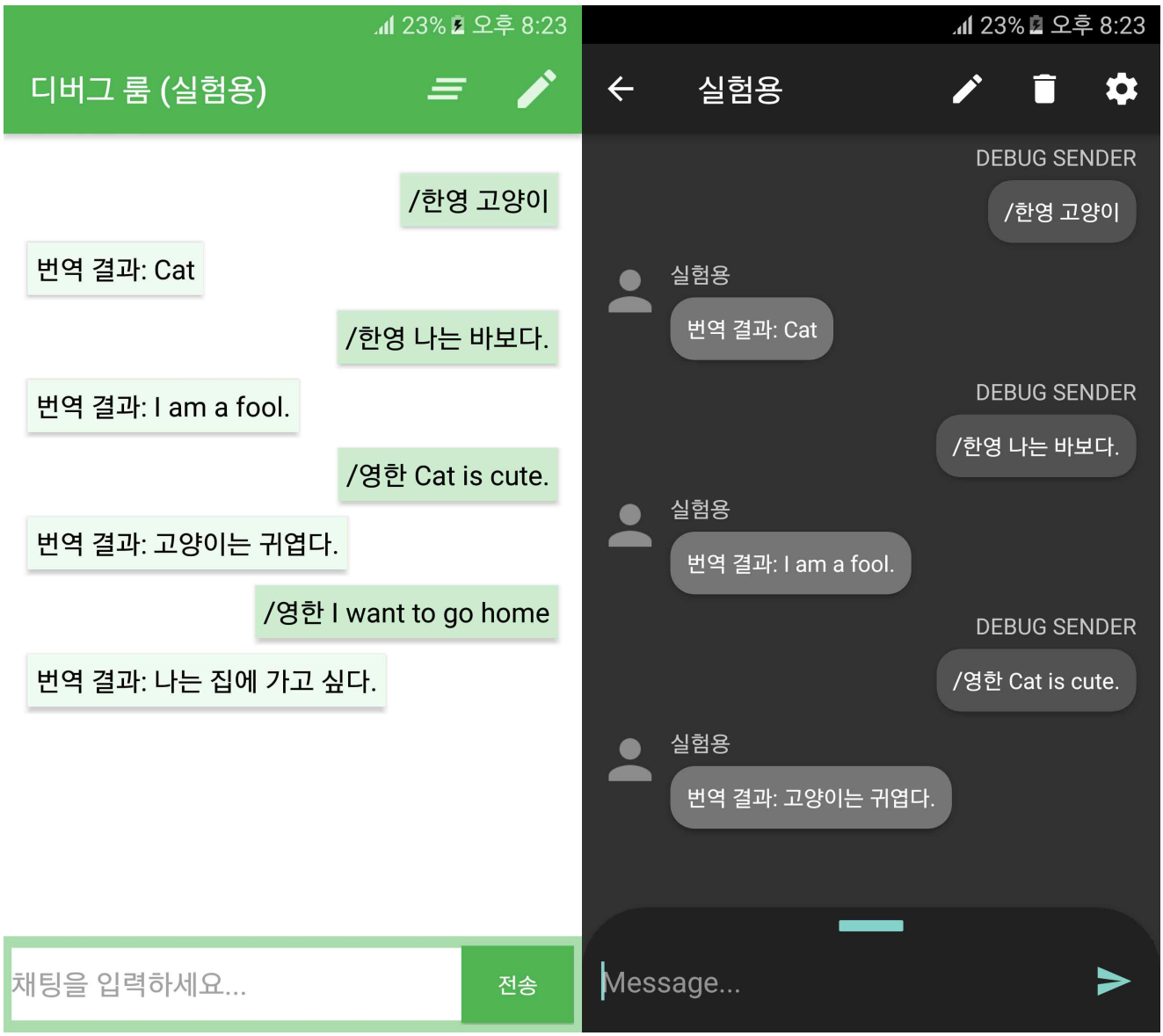
직접 번역기를 구현하는 것은 몹시 어렵기 때문에, 다른 곳에서 구현한 번역 API 등을 사용해야 해요. 하지만, 채팅 자동응답 봇과 메신저봇R 모두 Api.papagoTranslate();라는 번역을 해주는 함수를 지원하는지라, 그냥 저거 가져다가 쓸거에요.

3. 구현

```
1  function response(room, msg, sender, isGroupChat, replier) {
2      var cmd = msg.split(" ");
3
4      /* 수신된 채팅 내용 중 앞부분을 삭제 */
5      var data = msg.replace(cmd[0] + " ", "");
6
7      /* 자른 결과들 중 가장 앞에 있는게 '/영한'이라면, */
8      if (cmd[0]=="/영한") {
9
10         /* 영어 -> 한국어 번역 */
11         var result = Api.papagoTranslate("en", "ko", data);
12
13         /* 번역 결과 전송 */
14         replier.reply("번역 결과: "+result);
15
16     /* 1번째 if 종료 */
17     }
18
19     /* 자른 결과들 중 가장 앞에 있는게 '/한영'이라면, */
20     if (cmd[0]=="/한영") {
21
22         /* 한국어 -> 영어 번역 */
23         var result = Api.papagoTranslate("ko", "en", data);
24
25         /* 번역 결과 전송 */
```

```
25            /* 번역 결과 전송 */
26            replier.reply("번역 결과: "+result);
27
28        /* 2번째 if 종료 */
29        }
30
31    /* 이벤트 리스너 종료*/
32    }
```

4. 실행 모습



12. 현재 시간 출력

1. 작동 방식 구상

"/시간"이라는 채팅이 수신되면 현재 시간을 전송할거에요.

2. 필요한 것들

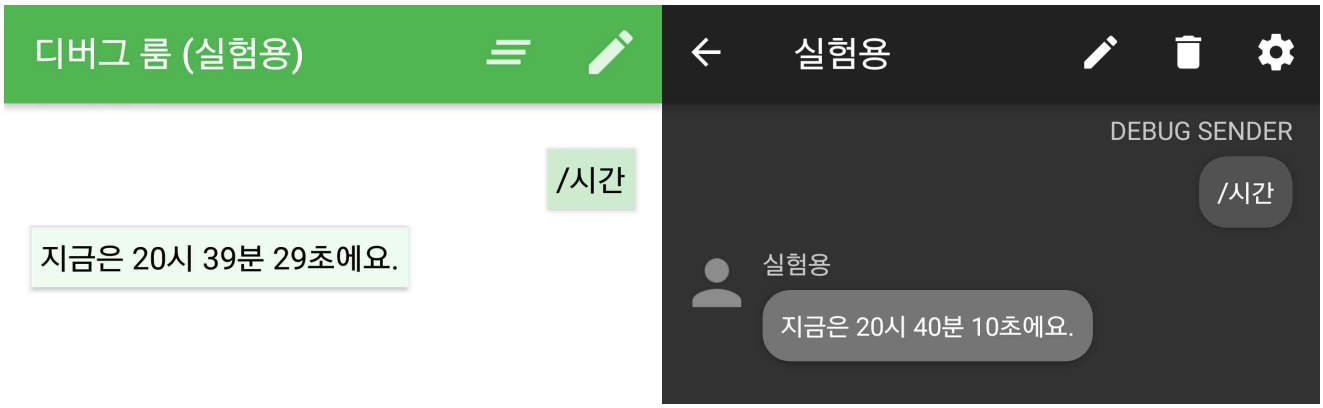
자바스크립트에서는 Date 객체를 사용하여 현재 시간 및 날짜를 가지고 올 수 있어요. 네2버나 9글에 '자바스크립트 시간' 등을 검색하시면 알 수 있어요.

3. 구현

5번째 줄은 4번째 줄이 너무 길어져서 다음 줄로 넘긴거에요. 굳이 따라하실 필요는 없는거에요. 하지만, 가끔씩은 다음 줄로 넘기는 것이 더 보기 좋을 때도 있는거에요.

```
1  function response(room, msg, sender, isGroupChat, replier) {
2      if (msg == "/시간") {
3          var day = new Date();
4          replier.reply("지금은 " + day.getHours() + "시 " + day.getMinutes() +
5              "분 " + day.getSeconds() + "초예요.");
6      }
7  }
8
```

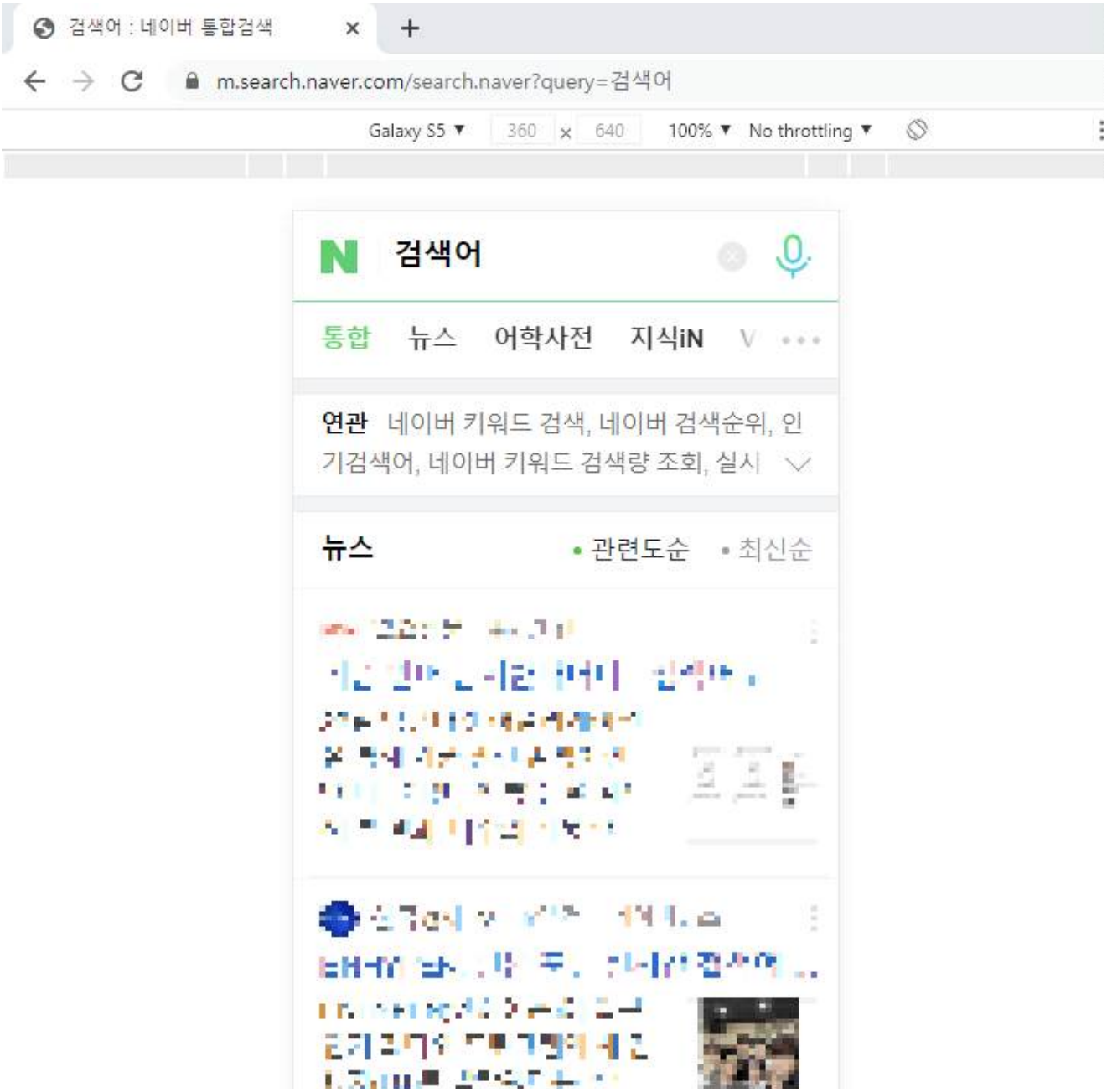
4. 실행 모습



13. 네이버 검색

1. 작동 방식 구상

아래 사진과 같이, 네이버에 검색하시고 url을 보면, 뒤에 검색어가 붙는거에요.



2. 필요한 것들

딱히 없어요. 그냥 저 https://m.search.naver.com/search.naver?query= 뒤에다가 검색할 내용을 붙이기만 하면 되는거에요. 다음이나 구글, YouTube 검색 기능도 같은 방식으로 만들 수 있어요.

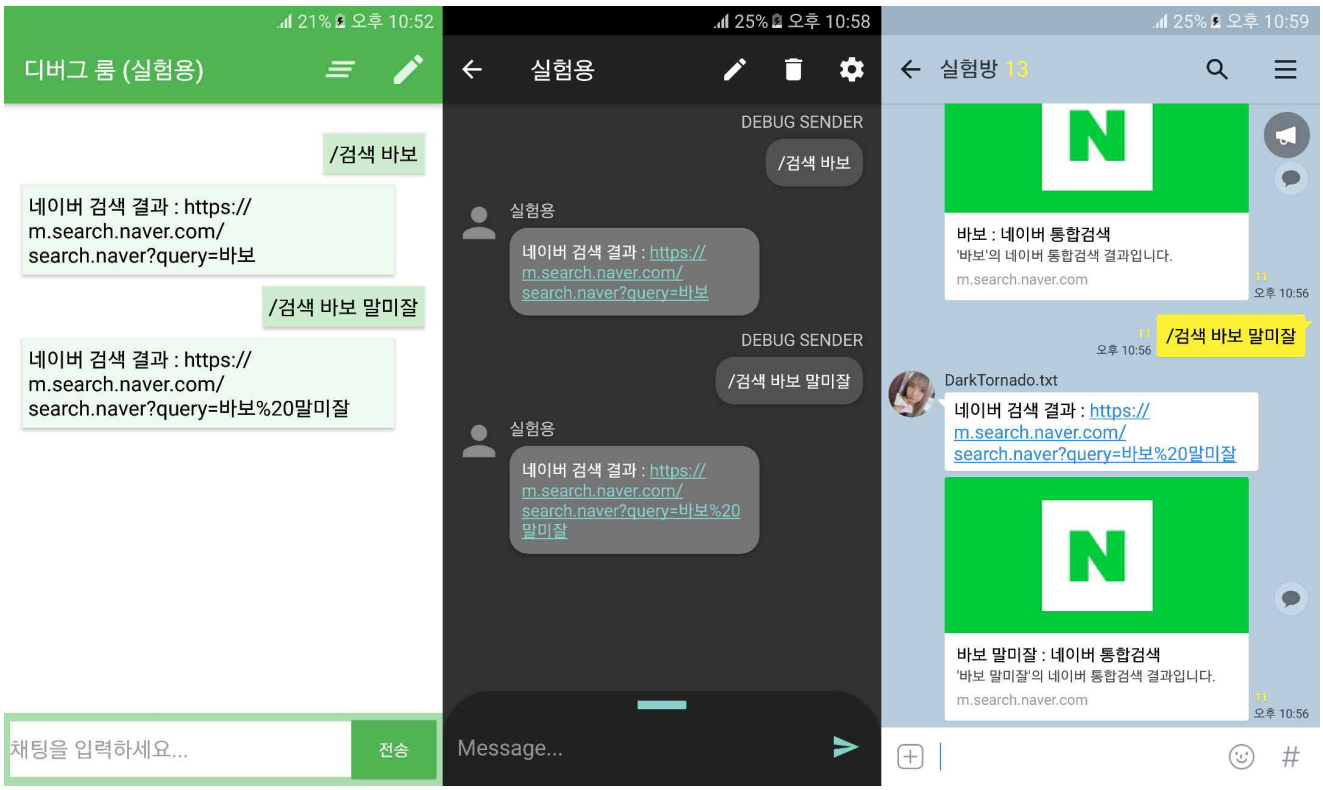
하지만, 띄어쓰기가 들어간 경우, url이 중간에 끊기는거에요. 그러니 모든 띄어쓰기를 %20으로 변경할거에요. url을 보면 %EB%B0%94%EB%B3%B4 같은 문자들이 보이지요? 그 중 %20가 띄어쓰기를 의미해요. 참고로, %EB%B0%94%EB%B3%B4는 바보에요.

3. 구현

```
function response(room, msg, sender, isGroupChat, replier) {
    var cmd = msg.split(" ")[0];
    var data = msg.replace(cmd + " ", "");
    if (cmd == "/검색") {
        replier.reply("네이버 검색 결과 : https://m.search.naver.com/search.naver?query="
            + data.replace(/ /g, "%20"));
    }
}
```

/ /g는 정규식이에요. 여기서는 모든 띄어쓰기를 찾기 위해 사용한거에요.

4. 실행 모습



14. 주사위

1. 작동 방식 구상

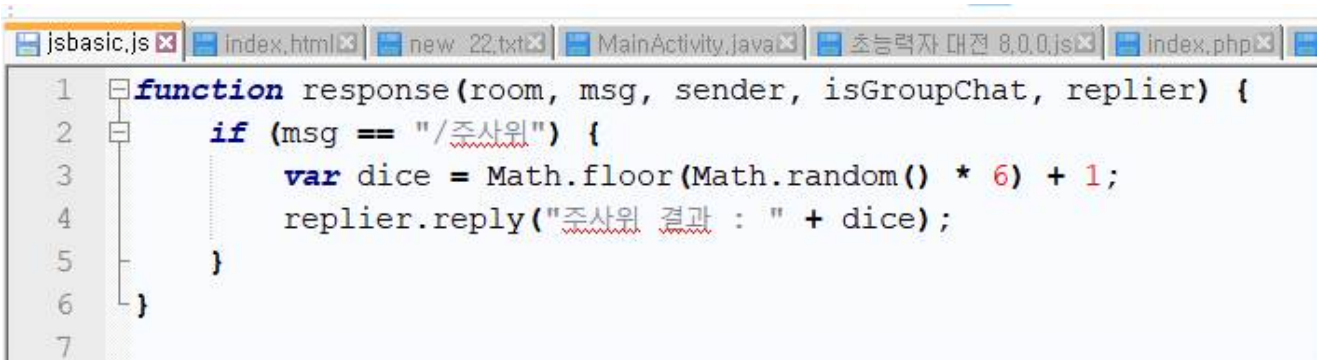
"/주사위"라는 채팅이 수신되면 1~6 사이의 정수를 랜덤으로 하나 골라서 출력할거에요.

2. 필요한 것들

포털 사이트에 "자바스크립트 랜덤'이나 "자바스크립트 난수" 등을 검색하면 Math.random();이라는 것이 존재한다는 것을 알 수 있을거에요. Math.random();은 0 이상 1 미만인 실수들 중 하나를 반환해요.

그러니, 그 결과에 6을 곱하면 0 이상 6 미만인 실수가 나올 것이고, 소수점 이하를 버리면 0 ~ 5 사이의 정수가 되고, 1을 더하면 1 ~ 6 중 하나가 랜덤으로 나오는 주사위 기능을 만들 수 있는거에요.

3. 구현



핵심 부분인 var dice = Math.floor(Math.random() * 6) + 1;를 보자면,

Math.random()
0 이상 1 미만인 실수 생성

Math.random() * 6
0 이상 6 미만인 실수 생성

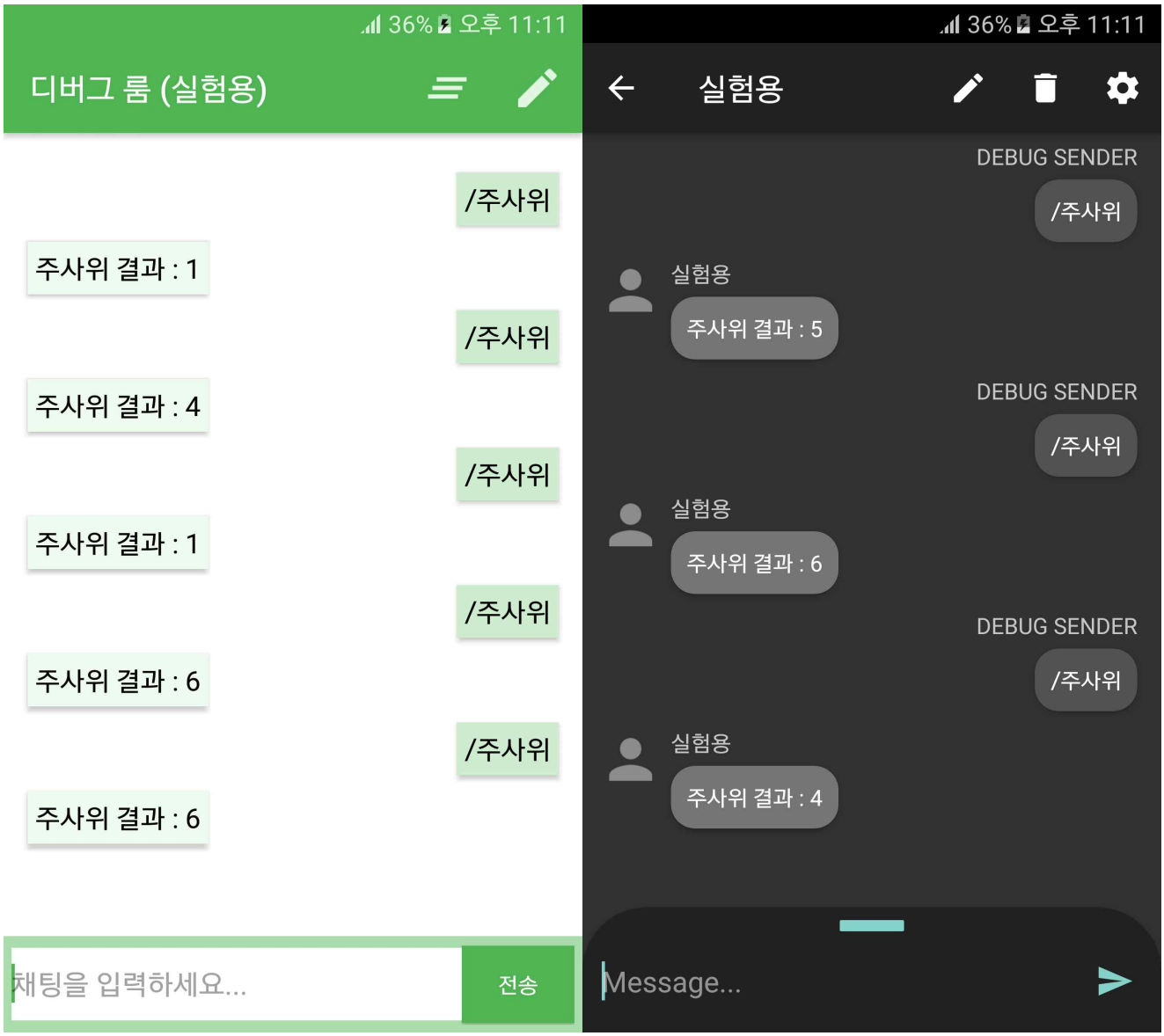
Math.floor(Math.random() * 6)
0 이상 6 이하인 정수 생성 (= 0 ~ 5 중 자연수 하나 랜덤 선택)

Math.floor(Math.random() * 6) + 1;
1 이상 6 이하인 정수 생성

var dice = Math.floor(Math.random() * 6) + 1;
그 결과를 변수 dice에 저장

인거에요.

4. 실행 모습



15. 로또

1. 작동 방식 구상

"/로또"라는 채팅이 수신되면 1 ~ 45 사이의 정수 중 7개를 겹치지 않게 랜덤으로 뽑아서 전송할거에요.

2. 필요한 것들

중복 방지를 위해 배열을 하나 만들어서 뽑은 수를 그 배열에 저장할거에요. 그러면 랜덤으로 뽑은 수가 배열에 있는지 없는지 확인하는 방법으로 중복 방지를 구현할 수 있어요. 배열에도 문자열처럼 .includes();가 있는거에요.

이런식으로, 어떻게 하면 될지 곰곰히 생각해보는 것이 좋은거에요. 나중에 프로그래밍 경험이 많아지면 어떻게 구현하면 될지 바로바로 생각할 수 있는 순간이 올거에요.

결과적으로는 뽑은 수들이 배열에 저장될 것이니, 배열에 있는 요소들을 모두 출력하면 되는 거에요. for문을 이용하여 일일이 출력하는 법도 있지만, 배열에 있는 .join(); 메소드를 사용하면 배열에 있는 요소들을 문자열로 합칠 수 있어요. 문자열에 있는 split();과 반대되는 거라고 생각하시면 편한거에요.

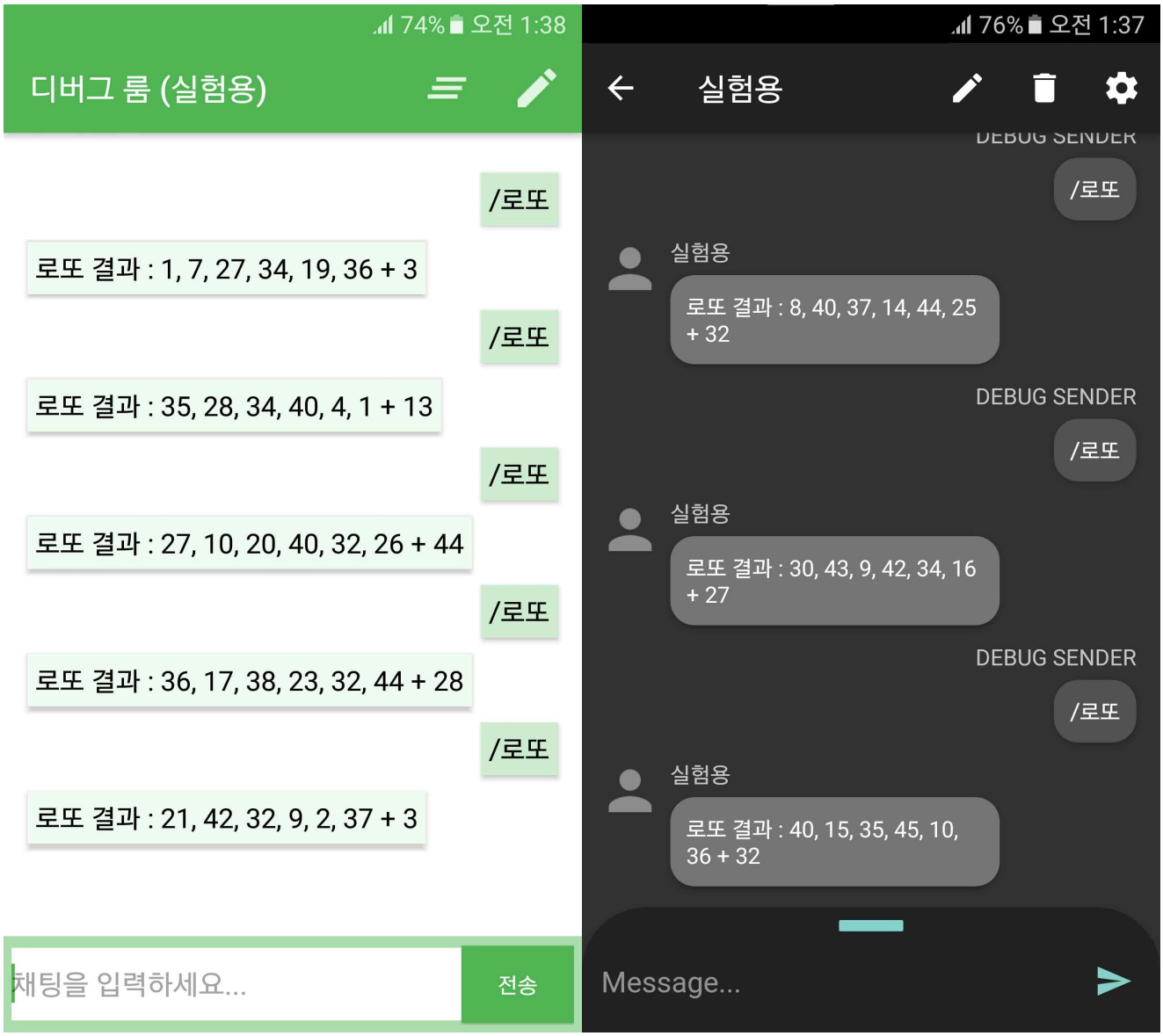
3. 구현

```
1  function response(room, msg, sender, isGroupChat, replier) {
2      if (msg == "/로또") {
3          var lotto = [];
4          for (var n = 0; n < 7; n++) {
5              var ran = Math.floor(Math.random() * 45) + 1;
6              if (lotto.includes(ran)) n--;
7              else lotto.push(ran);
8          }
9          var bonus = lotto.pop();
10         replier.reply("로또 결과 : " + lotto.join(", ") + " + " + bonus);
11     }
12 }
```

4. 설명

```
function response(room, msg, sender, isGroupChat, replier) {
이벤트 리스너 시작

if (msg == "/로또") {
수신된 채팅 내용이 "/로또"라면

var lotto = [];
선택된 수들이 들어갈 배열

for (var n = 0; n < 7; n++) {
보너스까지 7번 반복

var ran = Math.floor(Math.random() * 45) + 1;
1 이상 45 이하인 난수 생성

if (lotto.includes(ran)) n--;
이미 뽑힌 수라면 카운터 변수 1 감소. 따라서 1번 더 반복하게 됨

else lotto.push(ran);
아니라면 배열에 추가

}
for문 종료

var bonus = lotto.pop();
배열 중 가장 뒤에 있는 것을 따로 빼서 변수에 저장

replier.reply("로또 결과 : " + lotto.join(", ") + " + " + bonus);
결과 출력

}
if문 종료

}
function 종료
```

5. 실행 모습


74% 오전 1:38
디버그 룸 (실험용)
/로또
로또 결과 : 1, 7, 27, 34, 19, 36 + 3
/로또
로또 결과 : 35, 28, 34, 40, 4, 1 + 13
/로또
로또 결과 : 27, 10, 20, 40, 32, 26 + 44
/로또
로또 결과 : 36, 17, 38, 23, 32, 44 + 28
/로또
로또 결과 : 21, 42, 32, 9, 2, 37 + 3
채팅을 입력하세요...
전송
76% 오전 1:37
실험용
/로또
실험용
로또 결과 : 8, 40, 37, 14, 44, 25 + 32
DEBUG SENDER
/로또
실험용
로또 결과 : 30, 43, 9, 42, 34, 16 + 27
DEBUG SENDER
/로또
실험용
로또 결과 : 40, 15, 35, 45, 10, 36 + 32
Message...

16. 실제 로또 정보 확인

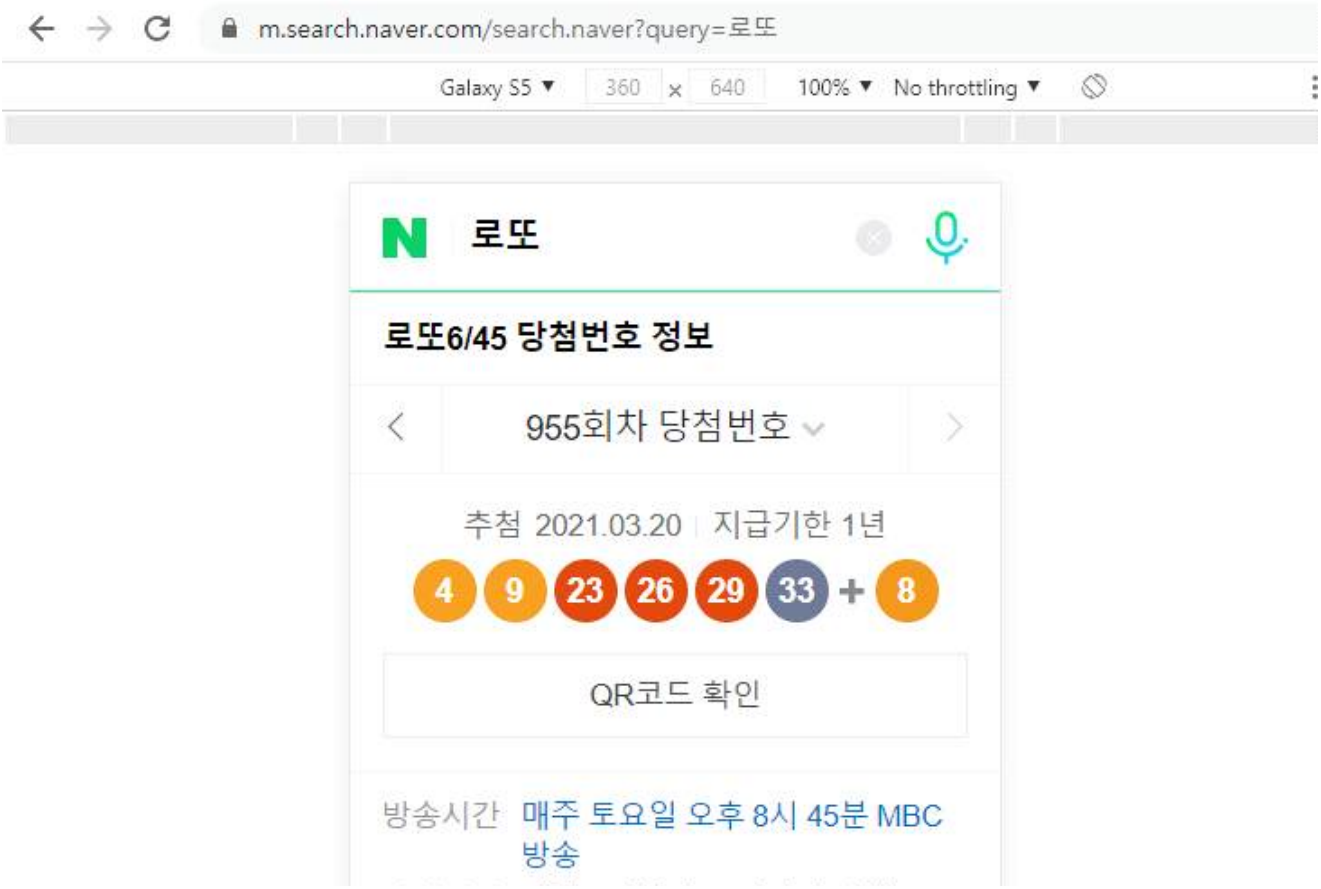
1. 작동 방식 구상

흔히 카톡봇 제작자들 사이에서 "파싱"이라고 불리는 것을 해볼거에요. 파싱보다는 크롤링에 더 가깝지만, 대부분 그냥 파싱이라고 부르는거에요.

네이버에 로또를 검색하면 가장 최근에 있었던 로또 번호가 뜨는거에요. 따라서, 그 부분을 긁어올꺼에요.

2. 필요한 것들

Utils.getWebText(); 함수를 통해 웹사이트의 내용을 가지고 올 수 있어요. 저기서 로또 번호만 남기고 다 지울거에요.



3. 구현

```
function response(room, msg, sender, isGroupChat, replier) {
    if (msg == "/로또") {
        var data = Utils.getWebText("https://m.search.naver.com/search.naver?query=로또");
        data = data.replace(/(<([^>]+)>)/g, "");
        data = data.split("1년")[2];
        data = data.split("QR")[0];
        data = data.trim();
        data = data.replace(/ /g, "");
        data = data.replace(/\n/g, " ");
        replier.reply("이번 회차 당첨 번호\n" + data);
    }
}
```

4. 설명

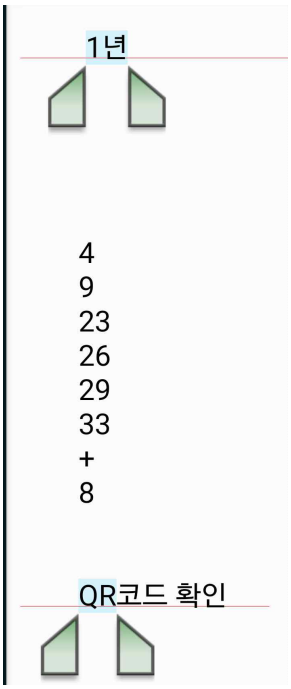
function response(room, msg, sender, isGroupChat, replier) {
채팅이 수신되면 작동

if (msg == "/로또") {
수신된 채팅 내용이 "/로또"라면,

var data = Utils.getWebText("https://m.search.naver.com/search.naver?query=로또");
네이버에 "로또"를 검색한 결과를 다 긁어와요.

data = data.replace(/(<([^>]+)>)/g, "");
HTML 태그 삭제. <이렇게> 생긴 것들을 전부 지운거에요.

data = data.split("1년")[2];
그 결과물에 중 1년이라는 문구가 두 개 있고, 두 번째 1년 뒤에 로또 번호가 있으니, 1년 단위로 자른 결과물 중 2번째를 가지고 오는거에요. 옆 이미지를 참고해보세요.



data = data.split("QR")[0];
잘린 결과물에 중 QR이라는 문구가 있고, 그 앞 부분에 로또 번호가 있으니, QR 단위로 자른 결과물 중 0번째를 가지고 오는거에요.

data = data.trim();
앞뒤에 붙은 불필요한 공백들을 모두 지워요. 옆 이미지를 참고해보세요.

data = data.replace(/ /g, "");
모든 공백들을 지워요

data = data.replace(/₩n/g, " ");
그리고, 엔터를 공백으로 바꿔요 ₩n는 엔터를 의미해요. ₩는 폰트에 따라 역슬래시로도 보이고 원 기호로도 보일거에요. 아무튼 원래는 역슬래시에요.

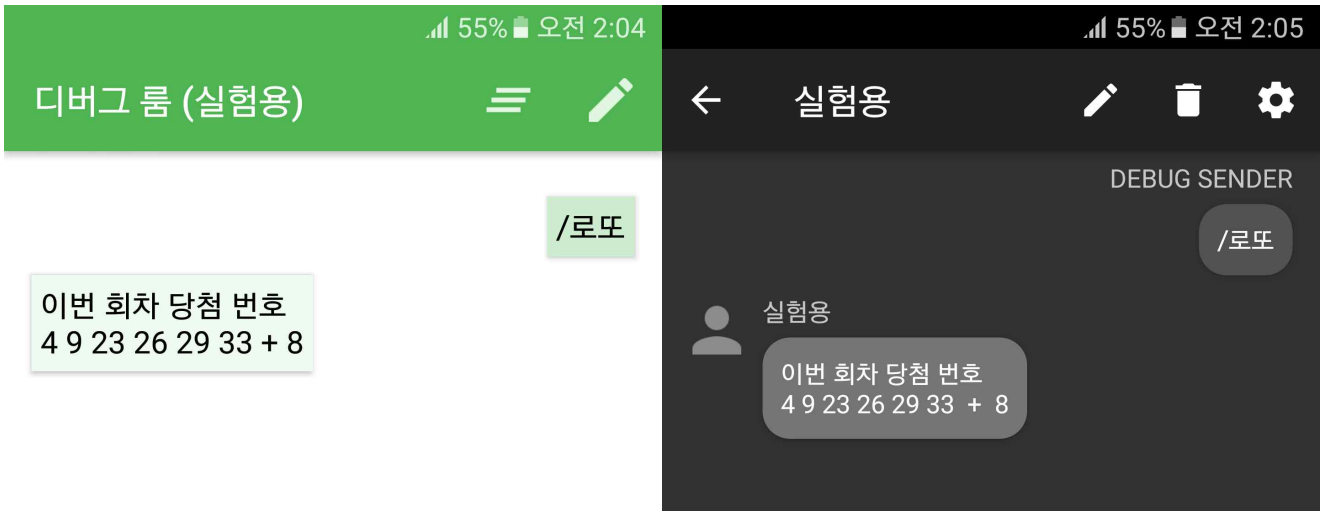
replier.reply("이번 회차 당첨 번호₩n" + data);
결과물을 출력해요

}
if문 종료

}
function 종료

5. 작동 모습

긁어온 결과에서는 엔터로 구분되어 있었지만, .replace(); 메소드를 통해 띄어쓰기로 바꾸었기에 이런식으로 출력되는거에요.



17. 실제 로또 정보 확인 2

1. 작동 방식 구상

웹사이트를 긁으라고 만들어진 라이브러리들이 있어요. 그 중 자바로 만들어진 jsoup이라는 라이브러리도 있는거에요. 1장 마지막 부분에서 설명한 내용과 같이, 비공식 카카오톡 봇에서는 jsoup 라이브러리를 사용할 수 있어요.

특정 조건을 가진 태그들을 선택하는 것이 가능한지라, 날로먹기 몹시 좋은거에요.

2. 필요한 것들

아무튼 jsoup 라이브러리로 날로먹을거에요.

3. 구현 및 설명

```
function response(room, msg, sender, isGroupChat, replier) {
    if (msg == "/로또") {
        var data = org.jsoup.Jsoup.connect("https://m.search.naver.com/search.naver?query=로또").get();
        data = data.select("div.lot_num");
        data = data.select("li");
        replier.reply("이번 회차 당첨 번호\n" + data.text());
    }
}
```

몹시 짧아진거에요. 핵심 부분인 if 내부만 설명하자면,

var data = org.jsoup.Jsoup.connect("https://m.search.naver.com/search.naver?query=로또").get();
일단 웹사이트 내용을 다 긁어온 것이고,

data = data.select("div.lot_num");

위 사진과 같이 class가 lot_num인 태그 속에 로또 번호들이 있으니, .select("태그.클래

스"); 메소드를 통해 저 영역만 선택한거에요.

```
data = data.select("li");
```

또, 위 사진과 같이, 각 수들은 <li> 태그 속에 있으니 .select("태그");를 통해 위에서 선택된 영역 속에서 <li> 태그들만 뽑은 뒤에,

```
replier.reply("이번 회차 당첨 번호₩n" + data.text());
```

.text();를 통해 태그들을 지워서 내용물만 출력한거에요.

4. 작동 모습

작동 모습은 이전 장에서 다룬 것과 동일하기에 생략하는거에요.

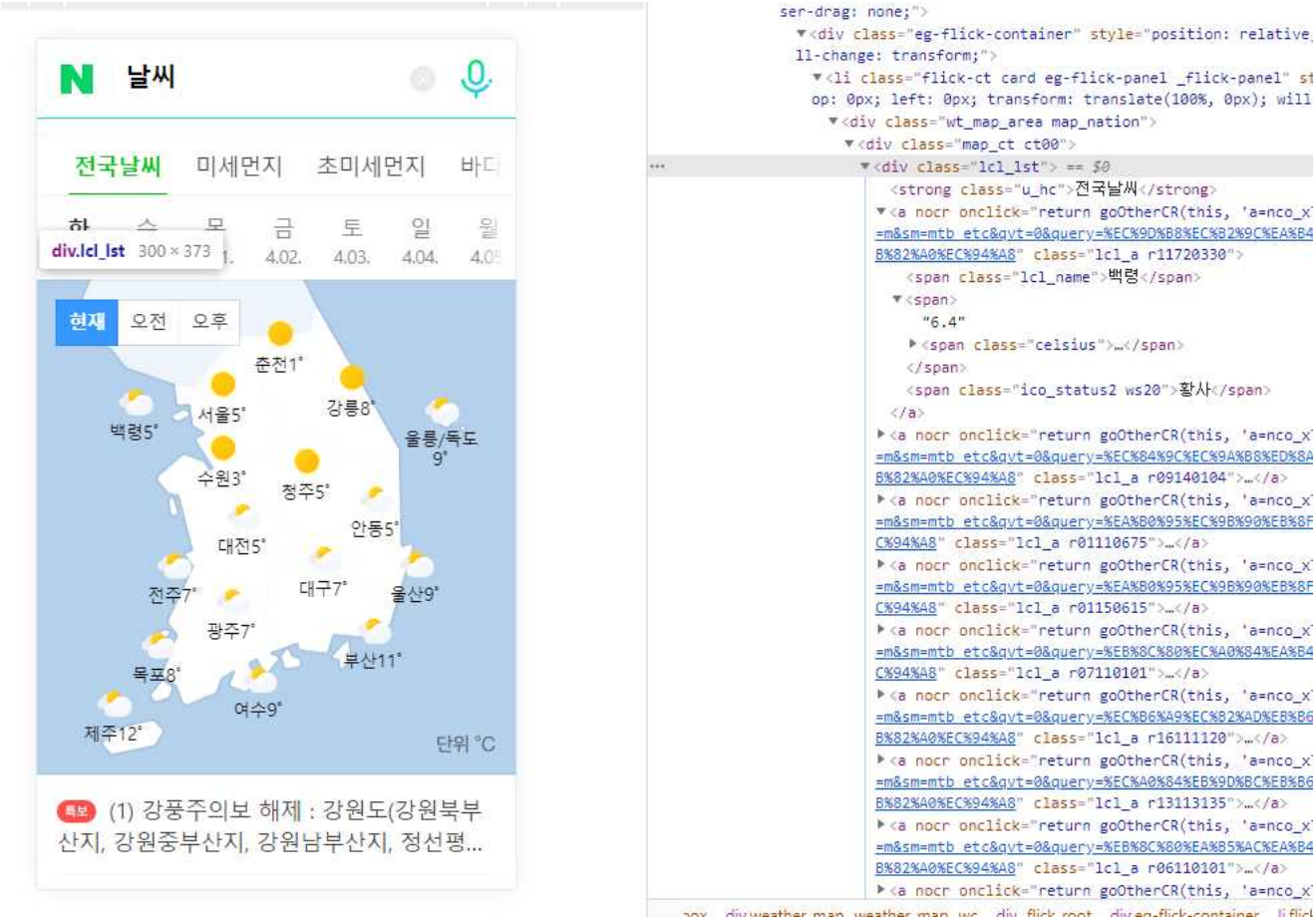
18. 전국 날씨

1. 작동 방식 구상

초록창에 날씨를 검색하면 전국 날씨 정보가 뜨니, 거기에서 날씨 정보만 뽑아올거에요.

2. 사이트 분석

class가 lcl_name인 태그들 중 0번째 태그 속 <a> 태그들에 각 지역의 날씨 정보가 있는거에요.



3. 구현 및 설명

```
function response(room, msg, sender, isGroupChat, replier) {
    if (msg == "/날씨") {
        var data = org.jsoup.Jsoup.connect("https://m.search.naver.com/search.naver?query=날씨").get();
        data = data.select("div.lcl_lst").get(0);
        data = data.select("a");
        var result = [];
        for (var n = 0; n < data.size(); n++) {
            result[n] = data.get(n).text();
        }
        replier.reply("[전국 날씨 정보]\n\n" + result.join("\n").replace(/도씨/g, "℃"));
    }
}
```

var data = org.jsoup.Jsoup.connect("https://m.search.naver.com/search.naver?query=날씨").get();
우선 네이버에 '날씨'를 검색한 내용을 다 긁어와요.

data = data.select("div.lcl_lst").get(0);
그 중 class가 lcl_lst인 태그들을 선택하고, 그 중 0번째 영역만 가지고와요

data = data.select("a");
그 영역 안에서 <a> 태그들만 골라요.

var result = [];
바로 .text();를 해버리면 날씨 정보들이 한 줄로 나오기에, 각 지역들의 날씨 정보를 배열에다가 넣어두고 나중에 각 요소들을 합칠 때 ₩n를 끼워넣을거에요

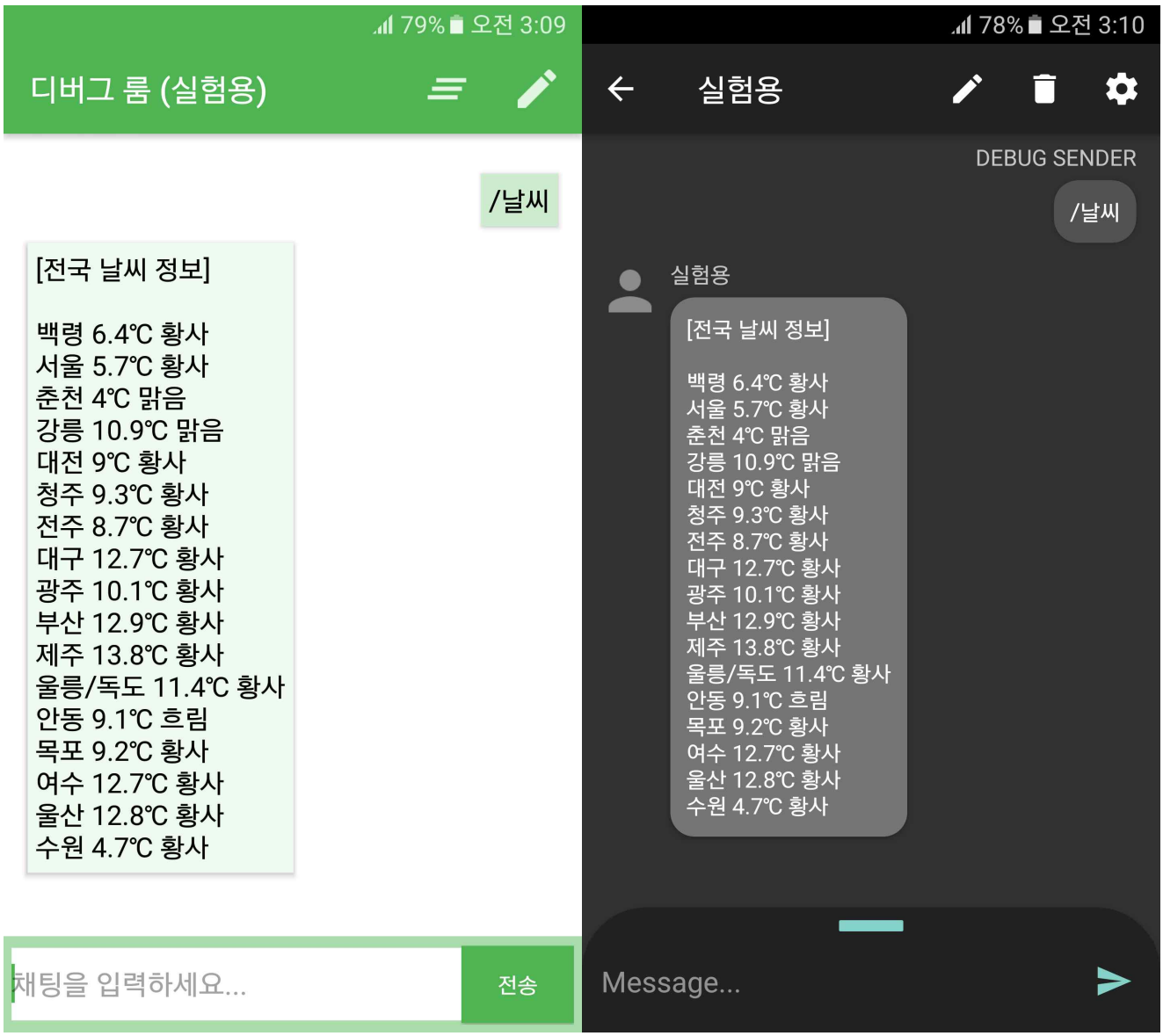
for (var n = 0; n < data.size(); n++) {
.select();로 선택한 영역의 자료형은 Elements이고, ArrayList를 상속받았기에, ArrayList에 있는 메소드들을 사용할 수 있어요.
배열이랑 비슷한건데, 길이를 가지고 올 때 .length 속성 대신 .size(); 메소드를 사용한다고 보시면 되는거에요.

result[n] = data.get(n).text();
이거도 이름[인덱스] 대신 이름.get(인덱스)라고 보시면 되는거에요.
각 a 태그들을 가지고와서 html 태그들을 제거한 결과물을 배열 result의 n번째 요소에 넣은거에요

}
for문 종료

replier.reply("[전국 날씨 정보]₩n₩n" + result.join("₩n").replace(/도씨/g, "℃"));
결과물 출력. join(); 메소드를 통해 배열을 문자열로 만들고, 개인적으로 "도씨"라는 문구가 뜨는게 마음에 들지 않아서 .replace(); 메소드를 통해 ℃로 바꾼거에요.

4. 작동 모습

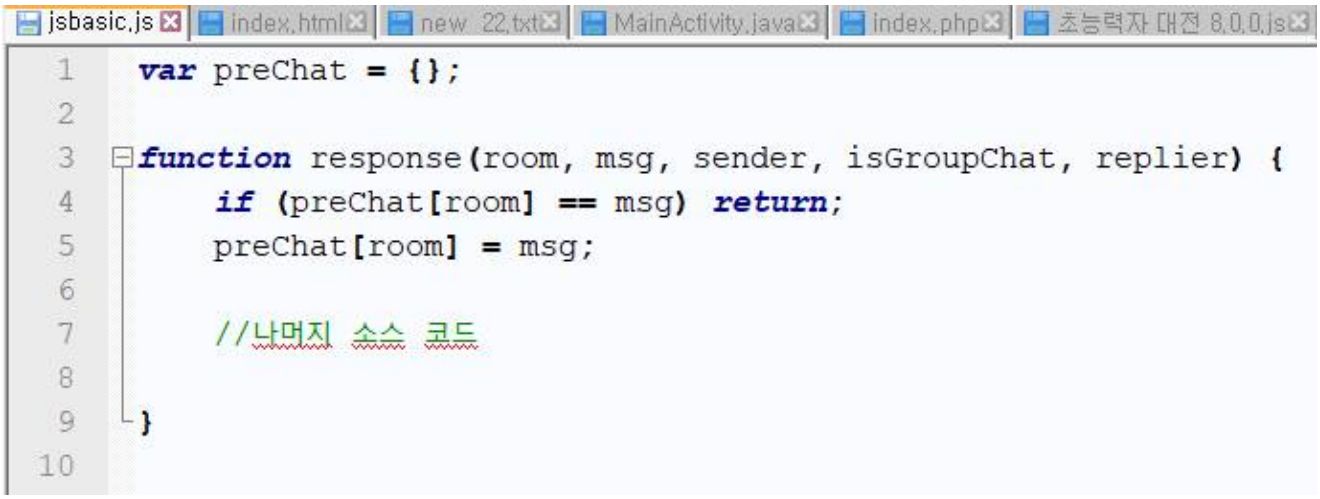

79% 오전 3:09
디버그 룸 (실험용)
/날씨
[전국 날씨 정보]
백령 6.4℃ 황사
서울 5.7℃ 황사
춘천 4℃ 맑음
강릉 10.9℃ 맑음
대전 9℃ 황사
청주 9.3℃ 황사
전주 8.7℃ 황사
대구 12.7℃ 황사
광주 10.1℃ 황사
부산 12.9℃ 황사
제주 13.8℃ 황사
울릉/독도 11.4℃ 황사
안동 9.1℃ 흐림
목포 9.2℃ 황사
여수 12.7℃ 황사
울산 12.8℃ 황사
수원 4.7℃ 황사
채팅을 입력하세요...
전송
78% 오전 3:10
실험용
DEBUG SENDER
/날씨
실험용
[전국 날씨 정보]
백령 6.4℃ 황사
서울 5.7℃ 황사
춘천 4℃ 맑음
강릉 10.9℃ 맑음
대전 9℃ 황사
청주 9.3℃ 황사
전주 8.7℃ 황사
대구 12.7℃ 황사
광주 10.1℃ 황사
부산 12.9℃ 황사
제주 13.8℃ 황사
울릉/독도 11.4℃ 황사
안동 9.1℃ 흐림
목포 9.2℃ 황사
여수 12.7℃ 황사
울산 12.8℃ 황사
수원 4.7℃ 황사
Message...

19. 도배 방지

1. 작동 방식 구상

같은 채팅이 연속으로 수신되면 가볍게 무시할거에요. 채팅방마다 따로따로 작동하도록 만들거에요. 이전에 수신된 채팅 내용을 저장해두었다가, 채팅이 수신될 때 마다 저장한 내용과 비교하여 무시할지 말지 정하는거에요.

2. 구현 및 설명



var preChat = {};
직전 채팅 내용이 저장될 객체

function response(room, msg, sender, isGroupChat, replier) {

if (preChat[room] == msg) return;
이미 저장되어 있는 채팅 내용(직전에 수신된 채팅 내용)과 지금 수신된 채팅 내용이 같다면 return을 통해 바로 function response 종료.
return은 원래 값을 반환하는 용도지만, 함수를 멈추면서 값을 반환하는 것이기에, return만 적어서 중간에 멈추게 만든거에요.

preChat[room] = msg;
직전 채팅 내용 업데이트

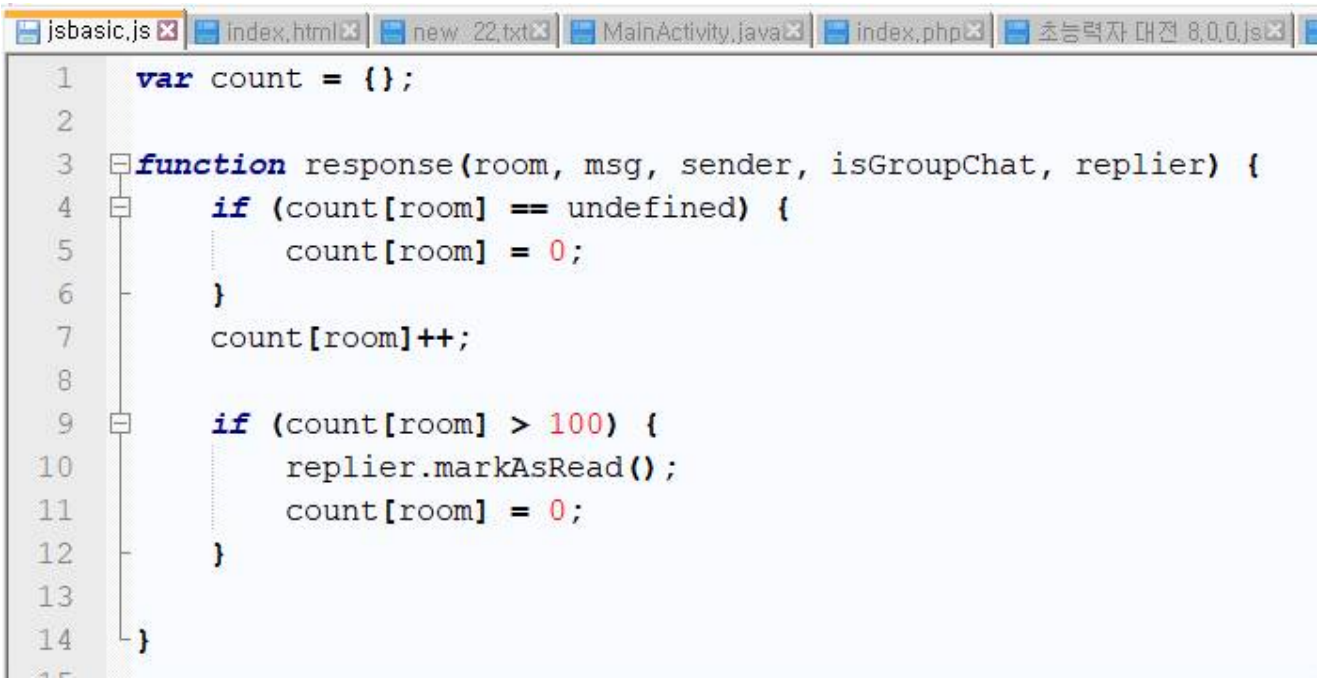
}
function 끝

20. 자동 읽음 처리

1. 작동 방식 구상

비교적 최근에 replier 인스턴스에 .markAsRead();라는 읽음으로 처리하는 메소드가 추가된거에요. 따라서, 저걸 사용하면 자동 읽음 처리가 가능한거에요. 채팅이 100개 수신될 때 마다 채팅을 자동으로 읽음 처리를 해볼거에요.

2. 구현 및 설명




```
var count = {};

function response(room, msg, sender, isGroupChat, replier) {
    if (count[room] == undefined) {
        count[room] = 0;
    }
    count[room]++;

    if (count[room] > 100) {
        replier.markAsRead();
        count[room] = 0;
    }

}
```

var count = {};
채팅이 수신된 횟수를 방마다 따로따로 세기 위해서 객체 사용

function response(room, msg, sender, isGroupChat, replier) {

if (count[room] == undefined) {
count[room] = 0;
}
해당 채팅방에서 채팅이 수신된 적이 없다면 0번으로 설정

count[room]++;
채팅 수신 횟수 1회 증가

```
if (count[room] > 100) {
수신된 채팅이 100개를 넘어가면

replier.markAsRead();
봇이 채팅을 보내지 않고 읽음으로 처리

count[room] = 0;
수신된 채팅 수 초기화

}

}
```

3. 기타 참고사항

일부 기기에서는 .markAsRead();를 사용할 수 없어요. 그리고, 이러한 기능들이 작동하는지 확인하기 위해서 채팅방에서 열심히 도배를 하면서 다른 사람에게 피해를 주는 경우가 종종 있으니, 자신과 봇 계정만 있는 채팅방에서 확인하거나, 테스트 할 때만 5개가 수신될 때 마다 읽음으로 처리하도록 바꾸는 것을 권장하는거에요.

21. POST 요청 보내기 - 파파고 API

1. 개요

앱 내장 API를 사용하지 않고, 파파고 API key를 발급받아서 jsoup 라이브러리를 통해 POST 요청을 보내볼거에요.

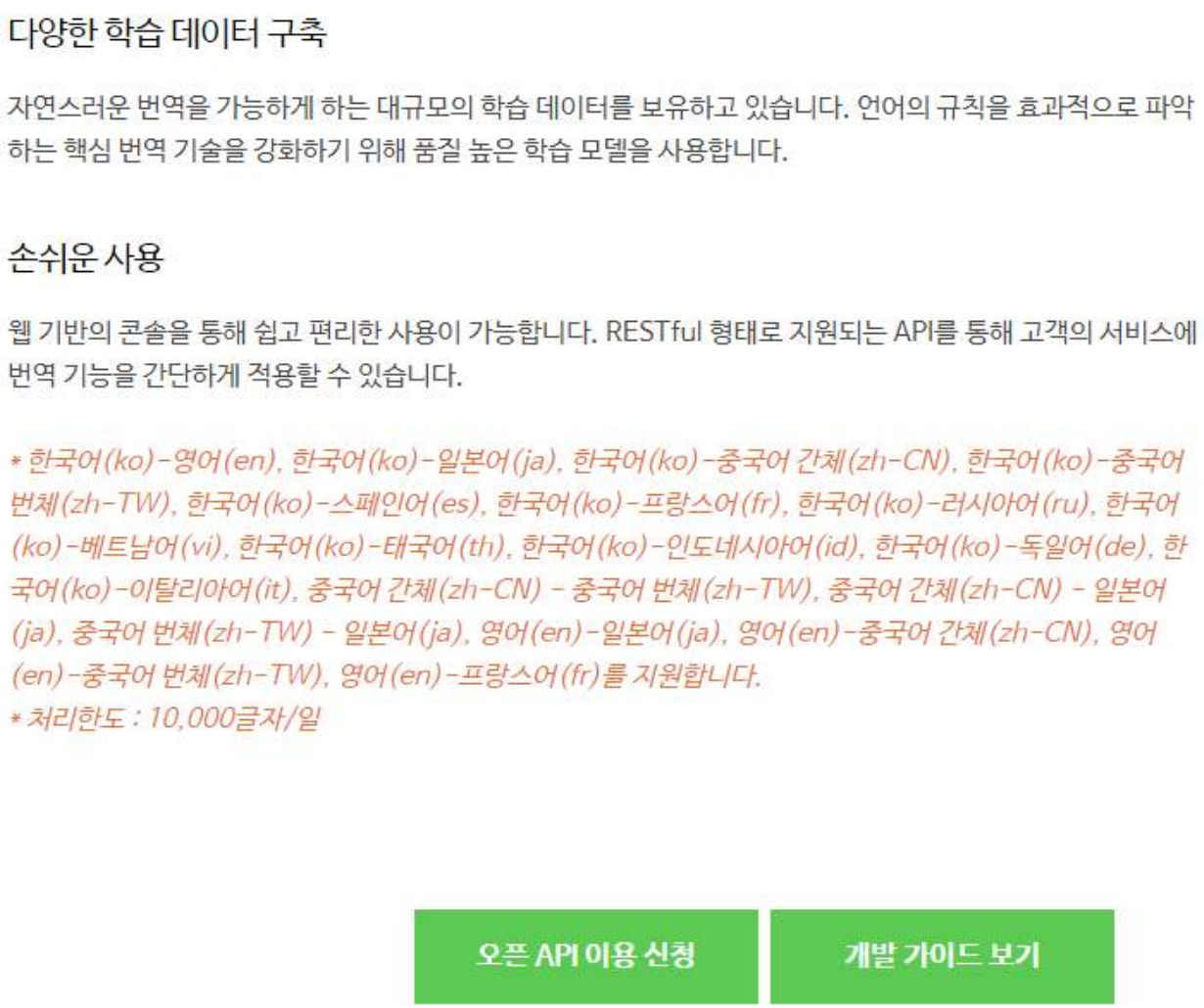
사실 위에서 jsoup을 통해 웹사이트의 내용을 긁어올 때 적었던 .get();이 GET 요청을 보내는 것이였던 거에요. 그 .get();만 .post();로 바꿔주면 POST 요청이 되는거에요.

2. API 키 발급받기

파파고 API를 사용하기 위해서는 API 키를 발급받아야 해요. 네이버 개발자 센터에 들어가면 파파고(papago)가 보일거에요. 그걸 누르시면 되는거에요.

https://developers.naver.com/main/

그러면 나오는 화면에서 "오픈 API 이용 신청 버튼"을 누르면 되는거에요. 본인인증을 하지 않은 계정으로 로그인을 하였다면, 본인인증 창이 뜰 수도 있어요.



이 이후는 네이버에서 잘 설명해주는거에요.
https://developers.naver.com/docs/papago/papago-nmt-overview.md

앱 이름은 능력것 입력하고, 서비스 환경은 Android로 설정하면 되는거에요. 그리고, Client ID와 Client Secret가 유출되지 않도록 살 관리해야 해요.

3. API 문서 읽어보기

어디로 어떻게 값을 넘기면서 요청을 해야 하는지 역시 네이버에서 잘 설명해주는거에요.
https://developers.naver.com/docs/papago/papago-nmt-api-reference.md

https://openapi.naver.com/v1/papago/n2mt로 HTTP 요청을 보내면 되고,

요청 URL

```
https://openapi.naver.com/v1/papago/n2mt
```

요청 방식은 POST이며,

HTTP 메서드

POST

파라미터는 다음과 같아요.

파라미터

파라미터	타입	필수 여부	설명
source	String	Y	원본 언어(source language)의 언어 코드
target	String	Y	목적 언어(target language)의 언어 코드
text	String	Y	번역할 텍스트. 1회 호출 시 최대 5,000자까지 번역할 수 있습니다.

발급받았던 키들은 헤더로 넘겨야 해요.

참고 사항

API를 요청할 때 다음 예와 같이 HTTP 요청 헤더에 **클라이언트 아이디와 클라이언트 시크릿**을 추가해야 합니다.

```
POST /v1/papago/n2mt HTTP/1.1
HOST: openapi.naver.com
User-Agent: curl/7.49.1
Accept: */*
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Naver-Client-Id: {애플리케이션 등록 시 발급받은 클라이언트 아이디 값}
X-Naver-Client-Secret: {애플리케이션 등록 시 발급받은 클라이언트 시크릿 값}
Content-Length: 51
```

요청 예

4. 실제로 POST 요청 보내기

```
function papagoTranslate(lang1, lang2, value) {
    var res = org.jsoup.Jsoup.connect("https://openapi.naver.com/v1/papago/n2mt")
        .header("X-Naver-Client-Id", "Client Id 입력") //헤더는 .header()로 보낼 수 있고
        .header("X-Naver-Client-Secret", "Client Secret 입력")
        .data("source", lang1) //파라미터는 .date()로 보낼 수 있어요
        .data("target", lang2)
        .data("text", value)
        .ignoreContentType(true)
        .ignoreHttpErrors(true)
        .post().text();
    return res;
}

function response(room, msg, sender, isGroupChat, replier) {
    if (msg == "테스트") {
        var result = papagoTranslate("en", "ko", "cat");
        replier.reply(result);
    }
}
```

아래와 같이 JSON 형식 문자열이 반환되는거에요.

테스트

{"message":{"@type":"response","@service":"naverservice.nmt.proxy","@version":"1.0.0","result":{"srcLangType":"en","tarLangType":"ko","translatedText":"고양이","engineType":"N2MT","pivot":null}}}

번역 결과는 message 객체 속 result 객체 속 translatedText 속에 들어있네요.

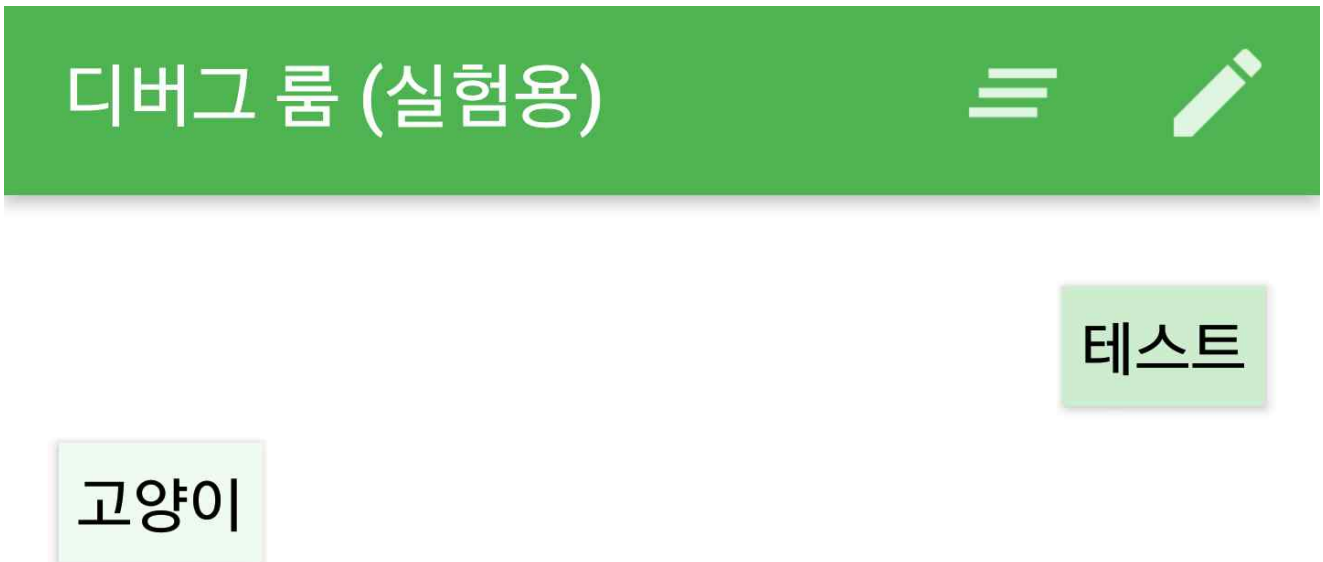
```
{
    "message": {
        "@type": "response",
        "@service": "naverservice.nmt.proxy",
        "@version": "1.0.0",
        "result": {
            "srcLangType": "en",
            "tarLangType": "ko",
            "translatedText": "고양이",
            "engineType": "N2MT",
            "pivot": null
        }
    }
}
```

그러니, 객체로 바꾸고 해당 위치에 접근한 결과를 반환하면 되는거에요.

jsbasic.js | index.html | new 22.txt | MainActivity.java | 초능력자 대전 8.0.0.js | Texia.txt | Dark Cheater 1.0.js | Dark Cheater

```
function papagoTranslate(lang1, lang2, value) {
    try {
        var res = org.jsoup.Jsoup.connect("https://openapi.naver.com/v1/papago/n2mt")
        .header("X-Naver-Client-Id", "Client Id 입력") //헤더는 .header()로 보낼 수 있고,
        .header("X-Naver-Client-Secret", "Client Secret 입력")
            .data("source", lang1)  //파라미터는 .data();로 보낼 수 있어요
            .data("target", lang2)
            .data("text", value)
            .ignoreContentType(true)
            .ignoreHttpErrors(true)
            .post().text();
        var data = JSON.parse(res);  //JSON 형식으로 된 문자열을 객체로 바꿔요
        return data.message.result.translatedText;  //해당 필드에 저장된 값을 반환해요
    } catch (e) {
        return null;  //위 과정에서 오류가 발생하면 null을 반환해요
    }
}

function response(room, msg, sender, isGroupChat, replier) {
    if (msg == "/테스트") {
        var result = papagoTranslate("en", "ko", "cat");
        replier.reply(result);
    }
}
```

이런식으로 번역이 잘 되는거에요.



4. 마무리

'/번역 [언어코드1] [언어코드2] [내용]'과 같은 형식으로 사용하도록 만든거에요.

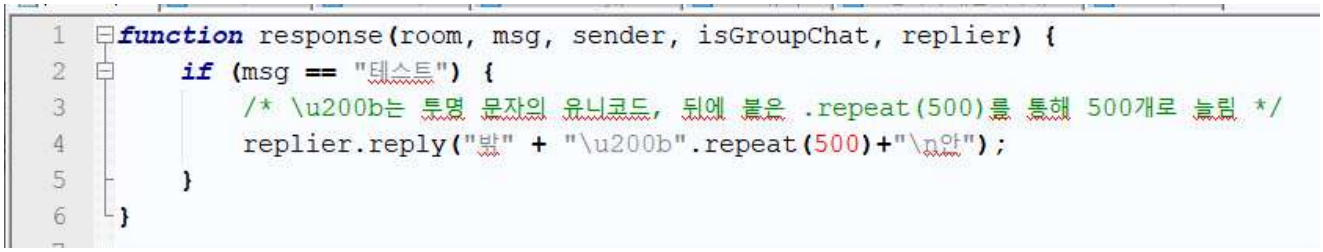
```
function papagoTranslate(lang1, lang2, value) {
    try {
        var res = org.jsoup.Jsoup.connect("https://openapi.naver.com/v1/papago/n2mt")
        .header("X-Naver-Client-Id", "Client Id 입력")
        .header("X-Naver-Client-Secret", "Client Secret 입력")
            .data("source", lang1)
            .data("target", lang2)
            .data("text", value)
            .ignoreContentType(true)
            .ignoreHttpErrors(true)
            .post().text();
        var data = JSON.parse(res);
        return data.message.result.translatedText;
    } catch (e) {
        return null;
    }
}

function response(room, msg, sender, isGroupChat, replier) {
    var cmd = msg.split(" ");
    if (cmd[0] == "/번역") {
        cmd.shift();
        var result = papagoTranslate(cmd.shift(), cmd.shift(), cmd.join(" "));
        if (result == null) replier.reply("번역 실패");
        else replier.reply(result);
    }
}
```

번역에 실패하면 응답으로 오는 JSON의 구조가 달라져서 오류가 뜰 것인지라, 귀찮은 관계로 그냥 try-catch를 사용한거에요.

22. 기타 여러 가지 것들

1. 전체보기 만들기

최신 버전 기준으로, 수신된 채팅이 500자를 넘어가면 그 뒷부분이 전체보기로 들어가니, 중간에 투명문자 500를 끼워넣으면 다음과 같이 뒷부분을 전체보기로 넣을 수 있어요.



```
function response(room, msg, sender, isGroupChat, replier) {
    if (msg == "테스트") {
        /* \u200b는 투명 문자의 유니코드, 뒤에 붙은 .repeat(500)를 통해 500개로 늘림 */
        replier.reply("밥" + "\u200b".repeat(500)+"\n안");
    }
}
```

2. 다른 방으로 채팅 전송

replier.reply(방, 내용); 또는 Api.replyRoom(방, 내용);곽 같이 특정 방으로 채팅을 보내는 것을 지워하는거에요.

하지만, 봇 구동 앱의 채팅 전송 방식은, 채팅 전송이 아니라 이미 채팅이 수신된 곳으로 응답을 전송하는 것이기에, 채팅이 수신된 적이 있는 곳으로만 채팅을 보낼 수 있어요. 앱 자체에서 채팅방마다 그 채팅방에 해당하는 replier들을 저장해두었다가, Api.replyRoom(방, 내용);와 같은 것을 사용하면 해당 방에 해당하는 replier를 통해 채팅을 보내는 것이라고 생각하면 이해하기 쉬운거에요.

3. try-catch-finally

try-catch-finally는 예외 처리를 위해 사용하는거에요. catch 또는 finally는 생략할 수 있지만 둘 다 생략하는 것은 불가능해요. try만 사용하는 것은 불가능하다는 뜻이에요.

try 블록 내부 소스를 실행하다가 예외가 발생하면 catch 부분으로 넘어가는거에요. catch의 매개변수는 그 예외의 내용이에요. .lineNumber 속성을 통해 어디서 예외가 발생했는지 확인할 수 있어요. finally 부분은 예외가 발생하든 말든 실행되는 부분이에요.

```
try {
    //예외가 발생할 수도 있는 소스
} catch (e) {
    //예외 발생시 실행되는 부분. e는 예외 내용
} finally {
    //예외 발생 여부와 관계없이 실행할 부분, try 블록 내부와 스코프 동일
}

```

부록

부록

1. 카카오톡 봇 문서

https://kkotbot-docs.kro.kr/
봇 구동 앱들에서 지원하는 API에 대한 설명들이 잘 정리되어 있는 곳이에요.

2. 봇 구동 체크리스트

https://github.com/DarkTornado/KakaoTalkBot/blob/master/CheckList.md
봇 구동을 위해 필요하거나 해야 할 것, 하지 말아야 할 것 등을 정리해둔 곳이에요.

3. 깃허브

https://github.com/DarkTornado/BasicKbot
이 책에 있는 예제들의 소스 파일들을 받을 수 있는 곳이에요.

4. 집필자 연락처

블로그 : https://blog.naver.com/dt3141592
이메일 : dt3141592@naver.com

5. 출간일

초판 : 서기 2021년 4월 27일