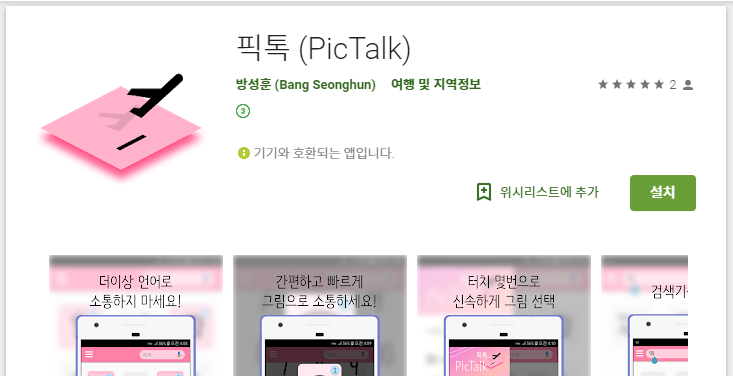
# 픽톡 (PicTalk)의 구현완성도

‘픽톡’팀의 구현완성도 문서입니다.

# ## 1. 제공 플렛폼

픽톡은 ‘**픽톡 (PicTalk)’**이라는 이름으로 **구글 플레이 스토어**에 출시한 상태입니다.



# ## 2. 핵심기술 코드 및 알고리즘

**- 음성인식**

코드1 :

**var** recognition = new SpeechRecognition();

recognition.lang = 'ko-KR';

recognition.interimResults = false;

recognition.maxAlternatives = 1;

*// var diagnostic = document.querySelector('.output');*

*// var bg = document.querySelector('html');*

*// var hints = document.querySelector('.hints');*

document.querySelector("#microphone").onclick = **function**() {

recognition.start();

console.log('Ready to receive a color command.');

}

recognition.onresult = **function**(event) {

*// The SpeechRecognitionEvent results property returns a SpeechRecognitionResultList object*

*// The SpeechRecognitionResultList object contains SpeechRecognitionResult objects.*

*// It has a getter so it can be accessed like an array*

*// The [last] returns the SpeechRecognitionResult at the last position.*

*// Each SpeechRecognitionResult object contains SpeechRecognitionAlternative objects that contain individual results.*

*// These also have getters so they can be accessed like arrays.*

*// The [0] returns the SpeechRecognitionAlternative at position 0.*

*// We then return the transcript property of the SpeechRecognitionAlternative object*

**var** last = event.results.length - 1;

**var** word = event.results[last][0].transcript;

*// diagnostic.textContent = 'Result received: ' + word + '.';*

console.log('Result received: ' + word + '.');

*// console.log('Confidence: ' + event.results[0][0].confidence);*

**var** replacedWord = word.replace(/\s/g, "");

document.querySelector("#search-bar").value = replacedWord;

search(replacedWord);

}

recognition.onspeechend = **function**() {

recognition.stop();

}

recognition.onnomatch = **function**(event) {

*// diagnostic.textContent = "I didn't recognise that color.";*

console.log("I didn't recognise that word.");

}

recognition.onerror = **function**(event) {

*// diagnostic.textContent = 'Error occurred in recognition: ' + event.error;*

console.log('Error occurred in recognition: ' + event.error);

}

코드2 :

<script type="text/javascript" src="cordova.js"></script>

기본적으로 음성인식 함수를 불러오고 옵션을 설정합니다.

옵션은 한국어를 불러오도록 맞춰놓고(recognition.lang), 음성인식을 했을 때 해당 음성의 추정되는 단어들을 2차원 배열로 불러오게 되는데 이때 가장 상위에 있는 한 개만 불러오도록 설정하면(recognition.maxAlternatives), 어차피 값을 한 개만 받아오기 때문에 단어별로 우선순위를 굳이 메길 필요가 없어집니다.(recognition.interimResults)

음성인식을 시작하는 이벤트가 발생할 경우 음성을 받기 위해 대기하며 음성이 들어오면 음성과 가장 처음 부합하는 단어를 선정하고 콘솔과 검색 필드에 해당 단어를 출력해줍니다. 음성이 한동안 입력되지 않으면 인식에 실패했음을 콘솔에 띄워주며 인식을 종료합니다. 이 상황들 이외에 다른 예상되지 못한 이벤트가 발생하면 콘솔에 해당 이벤트의 에러 값을 출력시켜줍니다.

픽톡의 음성인식은 인터넷에 연결되어있을 때 더 높은 정확도를 낼 수 있도록 만들었습니다.

인터넷에 연결되면 구글이 제공하는 웹 음성인식 표준 기술을 사용해 보다 높은 정확도로 반환할 수 있도록 기본적으로 자바스크립트 형태로 만들어져있습니다.

인터넷 연결이 끊겨있을 경우 안드로이드 자체에서 자바로 만들어진 음성인식 도구를 사용합니다. 기본적으로 코도바라는 폰갭과 자바스크립트로 만들어진 픽톡은 안드로이드에 있는 자바 코드를 바로 실행할 수 없습니다. 하지만 코도바에서 따로 제공하는 플러그인을 이용하면 자바스크립트에서 음성인식을 켰을 때 안드로이드 내장 자바 코드를 건드릴 수 있게끔 실행시켜줍니다.

**- 최근 선택, 자주 선택**

**var** historyPictogram = [];

**var** favoritePictogram = [];

historyPictogram = JSON.parse(localStorage.getItem("historyPictogram"));

favoritePictogram = JSON.parse(localStorage.getItem("favoritePictogram"));

if (historyPictogram == null) {

historyPictogram = [{id : "0"}];

favoritePictogram = [{id : "0", count : 0}];

}

앱이 실행될 때 로컬 스토리지에서 문자열로 변환된 history와 favorite 픽토그램을 불러와 다시 배열로 변환시켜 저장하지만 만약 앱을 처음 실행하여 로컬 스토리지가 비어있는 경우 더미 값을 집어넣어 줍니다.

*// 확대 버튼 누름*

fullScreenButton.addEventListener("click", **function** () {

**var** selecteds = document.querySelectorAll(".selected");

**var** selectedsStack = [];

**var** selectedsCount = 0;

for (**let** selected of selecteds) {

selectedsStack[selectedsCount] = selected;

selectedsCount++;

*// historyPictogram배열을 모두 검사하는데 historyPictogram.id와 selected.id가 같은게 아무것도 없으면 historyPictogram에 {id : selected.id}를 집어넣음*

if (historyPictogram.find(**function** (element) {

return element.id == selected.id;

}) == undefined) {

historyPictogram.unshift({

id : selected.id,

});

}

*// favoritePictogram.id와 selected.id가 다르다면 foundPictogram에 undefined를 저장하고 favoritePictogram에 {id : selected.id, count : 0,}를 추가함*

*// favoritePictogram.id와 selected.id가 같다면 foundPictogram에 selected.id를 저장하고 favoritePictogram에 count++ 해줌;*

**var** foundPictogram = favoritePictogram.find(**function** (element) {

return element.id == selected.id;

});

if (foundPictogram == undefined) {

favoritePictogram.push({

id : selected.id,

count : 0,

})

} else {

foundPictogram.count++;

}

}

historyPictogram = historyPictogram.slice(0, 12);

favoritePictogram.sort(**function** (a, b) {

return b["age"] - a["age"];

});

확대 버튼을 누르면 history 배열(최근 선택한 픽토그램을 담는 배열) 안에 있는 모든 값과 확대할 픽토그램이 같은지 검사하고, 어느 것과도 같지 않다면 history 배열 안에서 가장 오래된 픽토그램을 지우고 확대할 픽토그램을 ‘큐’ 구조로 다시 배열에 저장합니다.

favorite 배열(자주 선택한 픽토그램을 담는 배열) 역시 history 배열과 유사하게 배열 내의 모든 픽토그램과 확대할 픽토그램이 같은지 검사하고, 어느 것과도 같지 않으면 배열 안에 새로 추가하지만 기존에 있던 배열들은 지워지지 않습니다. 만약 같은 게 있다면 해당 픽토그램의 객체인 count의 값을 1 증가시킨 후 count의 값을 기준으로 배열 내의 위치를 정렬시켜줍니다.

localStorage.setItem("historyPictogram", JSON.stringify(historyPictogram));

localStorage.setItem("favoritePictogram", JSON.stringify(favoritePictogram));

선택한 픽토그램들을 모두 확대하고 나면 지금까지 변경되었던 배열의 정보를 로컬 스토리지에 다시 저장해줍니다.

**- 검색**

*// 검색 함수*

**function** search(query) {

for (**let** pictogram of pictograms) {

*// 각 픽토그램이 가진 클레스를 문자열로 변환*

**var** pictogramClass = (pictogram.classList).toString();

*// 만약 문자열이 query를 포함하고있지 않다면*

if (pictogramClass.includes(query) != 1) {

*// 없에버림*

pictogram.classList.add("display-none");

} else {

pictogram.classList.remove("display-none");

}

}

}

**function** reset() {

for (**let** pictogram of pictograms) {

pictogram.classList.remove("display-none");

}

}

search 함수가 실행되면 모든 픽토그램의 클래스를 검사하고 매개변수로 들어온 값과 같은 픽토그램을 제외한 나머지 픽토그램을 모두 제거합니다.

reset 함수가 실행되면 가려졌던 모든 픽토그램이 다시 보이도록 표시해줍니다.

*// 마이크 누름*

(document.querySelector("#microphone")).addEventListener("click", **function**(event) {

*// form의 기본 동작을 막아줌*

*// event.preventDefault();*

**var** query = document.querySelector("#search-bar").value;

if (query == "") {

reset();

} else {

search(query);

}

})

마이크를 누르면 검색 필드의 값을 받아와서 비어있다면 reset 함수를 실행시키고 그렇지 않다면 search 함수에 값을 매개변수로 넘겨주며 실행시킵니다.

# ## 3. 서버 구성

픽톡은 해외나 외지에서 인터넷이 끊긴 상황에서도 이용할 수 있도록 서버를 이용하지 않으며 서버 없이 개발되었습니다.