Create your own Language-Drive Application

Utilizing Google Cloud Speech API & Node.JS

Alex Goodman

Senior Software Engineer - Axia Technology Partners

The Internet of Things is here and it's growing fast. Smart, connected devices like Alexa are already transforming our world and are rapidly challenging businesses to keep up with innovation. According to estimations by the McKinsey Global Institute, the IoT will have a total economic impact of up to $11 trillion by 2025.

Now is the time to get ahead of the game.

In this session, I will guide you through how to create a simple IoT, Alexa-like application.

I will walk you through the challenges I encountered while writing my very own speech-recognition-powered GIF generator. I will be using Google Cloud Speech API and Node.JS to power my platform.

TABLE OF CONTENTS

DOWNLOADING THE NECESSARY FILES

1. Utilize github to clone https://github.com/Alexgoodman7/voice-driven-gif-generator

SETTING UP THE NODE ENVIRONMENT

1. First create a new server or, if capable, install the node environment on your computer.
2. Run the following commands to install the environment: (Below is for CentOS 7.3), visit https://nodejs.org/en/download/package-manager/ for your install guide

```
curl --silent --location https://rpm.nodesource.com/setup_7.x | bash -

yum install -y nodejs
```

2. Utilize github to clone https://github.com/Alexgoodman7/voice-driven-gif-generator
3. Install the required node packages for the project

```
cd /path/to/node/project/

npm install
```

4. package.json lists all the dependencies and will all be installed with “npm install”

```
{
 "name": "voice-driven-gif-generator",
 "description": "Create your own Language-Driven Application Utilizing Google Cloud Speech API &
Node.JS",
 "main": "server.js",
 "repository": "https://github.com/Alexgoodman7/voice-driven-gif-generator",
 "license": "ISC",
 "dependencies": {
  "@google-cloud/language": "^0.10.5",
  "@google-cloud/speech": "^0.9.3",
  "angular": "^1.6.4",
  "body-parser": "~1.4.2",
  "bootstrap": "^3.3.7",
  "express": "^4.15.2",
  "giphy-api": "^1.2.6",
  "greenlock": "^2.1.15",
  "greenlock-express": "^2.0.11",
  "jquery": "^3.2.1",
  "method-override": "~2.0.2"
 }
}
```

STEP 1: SET UP A SIMPLE NODE/EXPRESS HELLO WORLD APP

1. Refer to https://scotch.io/tutorials/setting-up-a-mean-stack-single-page-application
2. Build server.js (step1/server.js)

```
// server.js

// modules ================================================
var express        = require('express');
var app            = express();
var bodyParser     = require('body-parser');
var methodOverride = require('method-override');

// configuration ==========================================

// set our port
var port = process.env.PORT || 8080;

// get all data/stuff of the body (POST) parameters
// parse application/json
app.use(bodyParser.json());

// parse application/vnd.api+json as json
app.use(bodyParser.json({ type: 'application/vnd.api+json' }));

// parse application/x-www-form-urlencoded
app.use(bodyParser.urlencoded({ extended: true }));

// override with the X-HTTP-Method-Override header in the request. simulate DELETE/PUT
app.use(methodOverride('X-HTTP-Method-Override'));

// set the static files location /public/img will be /img for users
app.use(express.static(__dirname + '/public'));
app.use('/node_modules', express.static(__dirname + '/../node_modules'));

// routes ==================================================
require('./app/routes')(app); // configure our routes

// start app ===============================================
// startup our app at http://localhost:8080
app.listen(port);

// shoutout to the user
console.log('Magic happens on port ' + port);

// expose app
exports = module.exports = app;
```

3. Build routes.js (step1/app/routes.js)

```
// routes.js

module.exports = function(app) {

        // frontend routes =========================================================
        // route to handle all angular requests

        app.get('/', function(req, res) {
                res.sendfile('./public/views/index.html'); // load our public/index.html file
        });

}
```

4. Build index.html (step1/public/views/index.html)

```
<!-- index.html -->

<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>Untitled Document</title>
</head>

<body>
Hello World!
</body>
</html>
```

5. Run the server with “node server.js” (step1/server.js)
6. If relevant, open the firewall for port 8080
7. Test the server by curling or going to http://server:8080

STEP 2: BUILD A SIMPLE ANGULAR/BOOTSTRAP APP

1. Pull the CSS and HTML from free template: https://v4-alpha.getbootstrap.com/examples/cover/
2. Also pull the CSS and microphone image from https://webaudiodemos.appspot.com/AudioRecorder/index.html
3. Put it in main.css– I made some minor modifications, see (step2/app/public/css/main.css)
4. Modify index.html from STEP 1 and from the free template

```
<!-- index.html -->

<!DOCTYPE html>
<html lang="en"><head>
    <title>Voice-Driven GIF Generator</title>

    <!-- Bootstrap core CSS -->
    <link type="text/css" href="./node_modules/bootstrap/dist/css/bootstrap.min.css" rel="stylesheet">

    <!-- Custom styles for this template -->
    <link href="../css/main.css" rel="stylesheet">
  </head>

  <body>

    <div class="site-wrapper" ng-app="myApp">

      <div class="site-wrapper-inner">

        <div class="cover-container">

           <div ng-controller="RecordingController" class="inner cover">
            <p class="lead">Click on the mic and generate a GIF!</p>
            <p class="lead" style="cursor:pointer;">
                                <img id="record" src="img/mic128.png" ng-class="{'recording': recordingStarted === true, '':
recordingStarted === false}" ng-click="toggleRecording()">
            </p>
          </div>

          <div class="mastfoot">
            <div class="inner">
              <p>Download source code at <a href="https://github.com/Alexgoodman7/voice-driven-gif-generator">GitHub</a></p>
            </div>
          </div>

        </div>

      </div>

    </div>

    <!-- Bootstrap core JavaScript
    ================================================== -->
    <!-- Placed at the end of the document so the pages load faster -->
    <script src="./node_modules/jquery/dist/jquery.min.js"></script>
    <script src="./node_modules/angular/angular.min.js"></script>
    <script src="./node_modules/bootstrap/dist/js/bootstrap.min.js"></script>

    <!-- javascript files -->
        <script src="../js/controllers/recording-controller.js"></script>


</body></html>
```

5. Build recording-controller.js (step2/public/js/controllers/recording-controller.js)

```
// recording-controller.js

var app = angular.module('myApp', []);

app.controller("RecordingController", RecordingController);

function RecordingController($scope) {

        $scope.recordingStarted = false;

        $scope.toggleRecording = function() {
                if($scope.recordingStarted) {
                        $scope.recordingStarted = false;
                } else {
                        $scope.recordingStarted = true;
                }
        }
}
```

6. Run the node server (step2/server.js)
7. Test the app by going to http://server:8080 and then clicking the mic button on and off

STEP 3: BUILD A WEB APP TO CAPTURE MICROPHONE AUDIO

1. Refer to https://webaudiodemos.appspot.com/AudioRecorder/index.html
2. I made a "frankenstein" angular/native javascript version of their example to gather and show live audio from a client's microphone. NOTE: You can create your own front end, the meat of this project is in the backend.
    a. index.html (step3/public/views/index.html)
    b. recording-controller.js (step3/public/js/controllers/recording-controller.js)
    c. recorder.js (step3/public/js/recorder.js)
    d. recorderWorker.js (step3/public/js/recorderWorker.js)
3. In order to request microphone audio in modern browsers, you have to use HTTPS, so we need to modify server.js (step3/server.js) to create an SSL cert and listen on port 443 – refer to https://git.daplie.com/Daplie/greenlock-express

```
// server.js

// modules =================================================
var express        = require('express');
var app            = express();
var bodyParser     = require('body-parser');
var methodOverride = require('method-override');

// configuration ===========================================

// get all data/stuff of the body (POST) parameters
// parse application/json
app.use(bodyParser.json({limit: '1gb'})); // set limit to 1gb to account for sending base64 audio

// parse application/vnd.api+json as json
app.use(bodyParser.json({ type: 'application/vnd.api+json' }));

// parse application/x-www-form-urlencoded
app.use(bodyParser.urlencoded({ extended: true }));

// override with the X-HTTP-Method-Override header in the request. simulate DELETE/PUT
app.use(methodOverride('X-HTTP-Method-Override'));

// routes ==================================================
require('./app/routes')(app); // configure our routes

// set the static files location /public/img will be /img for users
app.use(express.static(__dirname + '/public'));
app.use('/node_modules', express.static(__dirname + '/../node_modules'));

// start app ===============================================
require('greenlock-express').create({

  server: 'https://acme-v01.api.letsencrypt.org/directory'

, email: 'alexgoodman7@gmail.com' // change this to your email

, agreeTos: true

, approvedDomains: [ 'gifgenerator.alexgoodman.net' ] // change this to your domain

, app: app

, renewWithin: (91 * 24 * 60 * 60 * 1000)
, renewBy: (90 * 24 * 60 * 60 * 1000)

, debug: false
}).listen(80, 443);

// expose app
exports = module.exports = app;
```

4. If relevant, open the firewall for port 80 and 443
5. Run the node server (step3/server.js)
6. Test it by going to https://server see if it requests your microphone, click the microphone image and see if it displays an audio stream

STEP 4: POST AUDIO TO NODE API AND SAVE IT AS A LOCAL FILE

1. Modify recording-controller.js (step4/public/js/controllers/recording-controller.js) to post audio as a base64 encoded blob

```
function doneEncoding( blob ) {
        var audioFileUrl = Recorder.setupDownload( blob, "myRecording" + ((recIndex<10)?"0":"") + recIndex + ".wav" );
        recIndex++;

        /*
         * STEP 4: POST AUDIO TO NODE API AND SAVE IT AS A LOCAL FILE
         * REFERENCE: https://stackoverflow.com/questions/23986953/blob-saved-as-object-object-nodejs
         * REFERENCE: https://stackoverflow.com/questions/20045150/how-to-set-an-iframe-src-attribute-from-a-variable-in-angularjs
         */

        var size = blob.size;
        var type = blob.type;

        var reader = new FileReader();
        reader.readAsDataURL(blob);
        reader.addEventListener("loadend", function() {

                var dataUrl = reader.result;
                var base64 = dataUrl.split(',')[1];
                var mediaFile = {
                        fileUrl: audioFileUrl,
                        size: blob.size,
                        type: blob.type,
                        src: base64
                }
                $http.post('/api/submit', mediaFile)
                        .then(function(response) {
                                console.log(response);
                                $scope.gifReceived = true;
                                $scope.gifURL = $sce.trustAsResourceUrl(response.data.gif_url);
                        }
                )

        })
}
```

2. Add an endpoint to routes.js (step4/app/routes.js) to receive audio and store it as a file

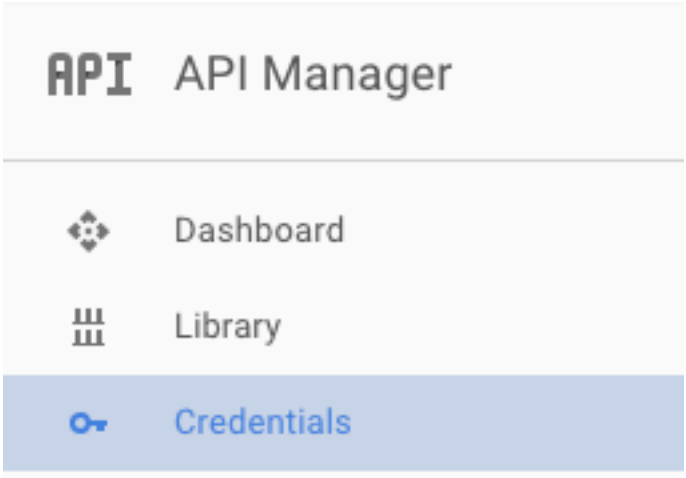
```
// routes.js

var fs = require('fs');

module.exports = function(app) {

        // route to handle creating goes here (app.post)

        app.post('/api/submit', function(req, res) {

                /*
                 * STEP 4: POST AUDIO TO NODE AND SAVE IT AS A LOCAL FILE
                 * REFERENCE: https://stackoverflow.com/questions/23986953/blob-saved-as-object-object-nodejs
                 */
                var original_response = res;

                //generate a unique filename
                var tmpFile = "/var/www/html/step4/public/files/"+Math.random().toString(36).substr(2)+".wav";

                //write recording to local file
                var src = req.body.src;
                var buf = new Buffer(src, 'base64'); // decode
                fs.writeFile(tmpFile, buf, function(err) {
                        if(err) {
                                console.log("err", err);
                        } else {
                                console.log(tmpFile);
                        }
                })
        })
```

3. Start your node server (step4/server.js)
4. Test this step by using the web app to record audio, the node console should output a file like "/var/www/html/step4/public/files/l6pxc09prdgaiou817ly58kt9.wav" – go to https://server/files/l6pxc09prdgaiou817ly58kt9.wav and verify it plays back the audio file

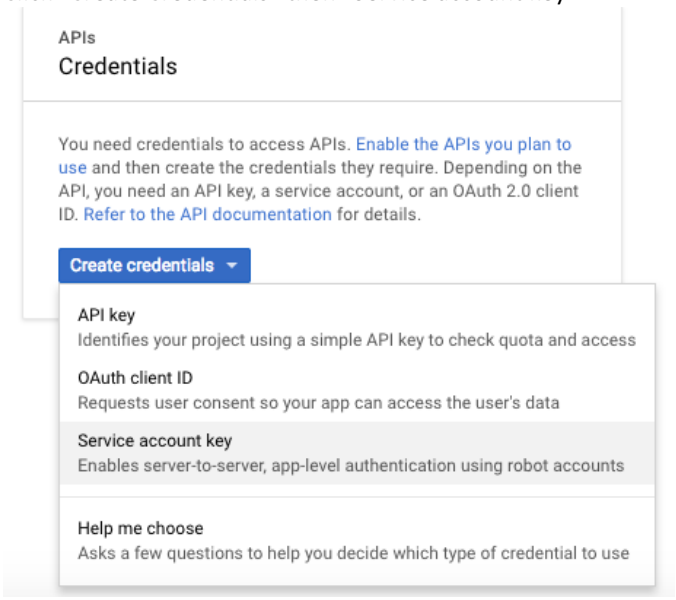
STEP 5: CONVERT AUDIO TO TEXT USING GOOGLE CLOUD SPEECH API

1. Create a Google Cloud Developer account- https://cloud.google.com/tools/docs/
2. Sign and click "TRY IT FREE"
3. Go through the prompts (you will have to put a Credit Card in, but it won't charge you)
4. Click on "My First Project" and create a new project
5. Click "API Manager" then "Enable API"
6. Click "Speech API"

7. Click "Enable"
8. Click "Credentials" tab

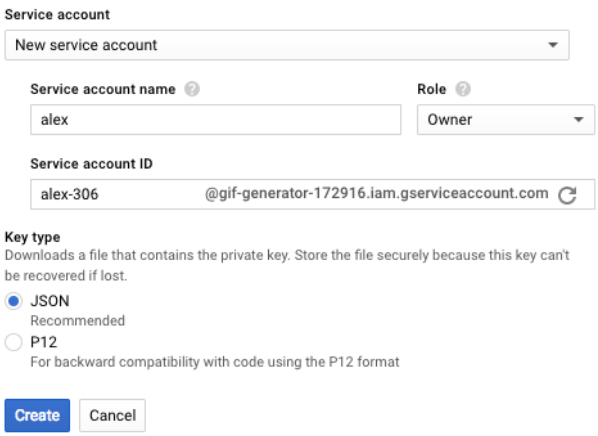


9. Click "Create Credentials" then "Service account key"



10. Click "Select…" under "Service account" then click "New service account"
11. Put in a "Service account name" and select "Role" as Project -> Owner

12. Make sure “Key type” is JSON then click “Create”



13. A key JSON file will be downloaded, save this to a local file on your server (not accessible by the outside)
14. Modify routes.js (step5/app/routes.js) to convert the audio to text using your Google Cloud key – refer to https://cloud.google.com/speech/docs/sync-recognize

```
/*
 * STEP 4: POST AUDIO TO NODE AND SAVE IT AS A LOCAL FILE
 * REFERENCE: https://stackoverflow.com/questions/23986953/blob-saved-as-object-object-nodejs
 */
var original_response = res;

//generate a unique filename
// in step 5 I change this to a tmp file so it's not publically accessible
var tmpFile = "/tmp/"+Math.random().toString(36).substr(2)+".wav";

//write recording to local file
var src = req.body.src;
var buf = new Buffer(src, 'base64'); // decode
fs.writeFile(tmpFile, buf, function(err) {
        if(err) {
                console.log("err", err);
        } else {
                console.log(tmpFile);

                /*
                 * STEP 5: CONVERT AUDIO TO TEXT USING GOOGLE CLOUD SPEECH API
                 *
                 * REFERENCES:
                 * https://cloud.google.com/speech/docs/sync-recognize
                 * https://www.npmjs.com/package/@google-cloud/speech
                 */

                // Imports the Google Cloud client library
                // Instantiates a client
                var speech = require('@google-cloud/speech')({
                  projectId: 'gif-generator-172623',
                  keyFilename: '/usr/local/gifgenerator/key.json' //replace this with your service account key location
                });

                // The path to the local file on which to perform speech recognition, e.g. /path/to/audio.raw
                const filename = tmpFile;

                // The encoding of the audio file, e.g. 'LINEAR16'
                const encoding = 'LINEAR16';

                // The sample rate of the audio file in hertz, e.g. 16000
                //I eliminated the need for this to account for different audio inputs
                //const sampleRateHertz = 44100;

                // The BCP-47 language code to use, e.g. 'en-US'
                const languageCode = 'en-US';

                const request = {
                  encoding: encoding,
                  languageCode: languageCode
                };

                // Detects speech in the audio file
                speech.recognize(filename, request)
                  .then((results) => {
                        const transcription = results[0];

                        console.log(`Transcription: ${transcription}`);
                })
        }
})
```

15. Start your node server (step5/server.js)
16. Test this step by recording your voice, and seeing the Transcript in the node console, it will look something like “Transcription: test”

STEP 6: USE GOOGLE NATURAL LANGUAGE API TO DETECT ENTITES IN TEXT

1. Login to Google Cloud Developer console, go to the API Manager
2. Click “Enable API”
3. Click “Natural Language API” then click “Enable”
4. Modify routes.js (step6/app/routes.js)

```
console.log(`Transcription: ${transcription}`);

/*
 * STEP 6: USE GOOGLE NATURAL LANGUAGE API TO DETECT ENTITIES IN TEXT
 *
 * REFERENCES:
 * https://cloud.google.com/natural-language/docs/analyzing-entities#language-entities-string-nodejs
 */

// Imports the Google Cloud client library
var language = require('@google-cloud/language')({
  projectId: 'gif-generator-172623', //replace this with your projectId
  keyFilename: '/usr/local/gifgenerator/key.json' //replace this with your service account key location
});

// The text to analyze, e.g. "Hello, world!"
var text = transcription;
if(text === "")
        text = "random"; // if there's no text detected, show a random gif

// Instantiates a Document, representing the provided text
const document = language.document({ content: text });

// Detects entities in the document
document.detectEntities()
  .then((results) => {
        const entities = results[1].entities;

        console.log('Entities:');
        var entities_string = '';
        entities.forEach((entity) => {
          console.log(entity.name);
                entities_string += entity.name + ' ';
          console.log(` - Type: ${entity.type}, Salience: ${entity.salience}`);
        });

        //if no entities found, just try to make a gif with the transcript
        if(entities_string === '')
                entities_string = text;

        console.log(entities_string);
})
```

5. Run your node server (step6/server.js)
6. Test this step by recording your voice, in the node console you will see the transcript, and information about the entities in the transcript

STEP 7: USE GIPHY API TO GRAB A GIF FROM THE DETERMINED ENTITIES

1. Modify routes.js (step7/app/routes.js) to use the giphy api to grab the entities and return a gif

```
console.log(entities_string);

/*
 * STEP 7: USE GIPHY API TO GRAB A GIF FROM THE DETERMINED ENTITIES
 *
 * REFERENCES:
 * https://www.npmjs.com/package/giphy-api
 */

var giphy = require('giphy-api')();

var giphy_data = {
        q: entities_string,
        limit: 1,
        rating: "pg"
}
giphy.search(giphy_data, function (err, res) {
        console.log(res);
        var url = res.data[0].embed_url;
        console.log(url);
        return original_response.json({'gif_url': url});
});
```

2. Front end already has the code from step 3 to receive the gif URL and show it
3. Run your node s erver (step7/server.js)
4. Test this step by recording your voice and seeing if a related gif shows up! You can also view the node console and see the response from the Giphy API

STEP 8: CHALLENGE YOURSELF!

1. After completing this project, you should try and push your project even further, good examples of extra things to do are:
    a. Stream Audio: instead of one request per recording, stream the audio and do silence detection to return gifs based on audio stream
    b. Make the gif generation smarter: Currently we return a gif based on a combination of all the entities mentioned. You could investigate other services to account for verbs, as well as looking at the "Salience" of each entity.
    c. Do cooler things than GIFs!
    d. Tie this into other APIs. Like: tell your IP lights to turn on using this tool!

REFERENCE INFORMATION

Cover Page: Created by Harryarts - Freepik.com

References:

https://scotch.io/tutorials/setting-up-a-mean-stack-single-page-application

https://v4-alpha.getbootstrap.com/examples/cover/

https://webaudiodemos.appspot.com/AudioRecorder/index.html

https://stackoverflow.com/questions/23986953/blob-saved-as-object-object-nodejs

https://cloud.google.com/speech/docs/sync-recognize

https://www.npmjs.com/package/@google-cloud/speech

https://cloud.google.com/natural-language/docs/analyzing-entities#language-entities-string-nodejs

https://www.npmjs.com/package/giphy-api

Contact me at:
alexgoodman7@gmail.com

Connect with me at LinkedIn:
http://alexgoodman.net