# Web Speech API – building online speech translator

*You can say ‘OK Google’ on your Android phone or tablet to start a voice search or have the device do something for you. iPhone and iPad users have Siri – and intelligent personal assistant that helps you get things done. Wouldn’t that be cool if we could navigate and interact with websites using voice commands as well?*

## What is Web Speech API?

This cutting edge technology introduced at the end of 2012 (<https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html#tts-section>) aims to enable web developers to provide, in a web browser, speech-input and text-to-speech output features beyond typical speech-recognition or screen-reader software. The Web Speech API is made of two main components:

* SpeechRecognition(<https://developer.mozilla.org/en-US/docs/Web/API/SpeechRecognition>) –provides the ability to check speech received through device’s microphone against a list of grammar(the vocabulary you want to be recognised in your app). List of words can be narrowed down using the SpeechGrammar(<https://developer.mozilla.org/en-US/docs/Web/API/SpeechGrammar>) interface. When a word or phrase is recognised, it is returned as a result that contains the phrase as a text string;
* SpeechSynthesis(<https://developer.mozilla.org/en-US/docs/Web/API/SpeechSynthesis>) – a text-to-speech component, which is pretty much self-explanatory - it allows web apps to read their content out loud.

You’re probably asking yourself ‘What about the browser support?’ As mentioned before, it is still new technology that hasn’t been implemented in all browsers. At the time of writing this tutorial (January 2017) SpeechRecognition is supported in Chrome, Opera and Chrome for Android with –webkit prefix and can be enabled in Firefor in config (<http://caniuse.com/#feat=speech-recognition>). On the bright side SpeechSynthesis made it to most of modern browsers (<http://caniuse.com/#feat=speech-synthesis>).

## Speech translator main features

Before starting any development it is good to list down functionality, so we can better plan and structure our code. The purpose of this tutorial is to show as many features of Web Speech API as possible, therefore the final app might look little exaggerated.

Let’s break down user’s journey into few points, so we can better imagine how the app is going to work.

1. User opens the app and changes source and target language using voice command
2. User recites what needs to be translated
3. The app outputs translation on the screen
4. User uses voice command to read the translation out loud by the browser
5. User uses voice command to display original speech recognised by the app
6. User changes text-to-speech speed using voice command
7. The app reads the text with new speed
8. User stops text-to-speech
9. User continues to recite for further translations

Now, when it’s clear what we’re going to build and how it’s going to work, let’s get our hands dirty, shall we?

## Setting up new project

An assumption is made that people who try to finish this tutorial have fundamental knowledge of CSS / SASS, HTML and JavaScript, hence we’re going to mainly focus on introduction of new API features and dedicate little attention to mark-up and styling.

For development I’m going to use SASS(add link), ES6(add link) and Gulp(add link) for more robust workflow. I’ve prepared a repository with starting files(add link to repo). If anyone prefers different setup, you’re welcome to use your tools of choice.

Let’s start with HTML file. We create a very simple interface that can be used alongside voice commands to use the app. Select tags don’t have any children on purpose, as those are going to be appended with JavaScript.

(html markup)

Next, add styles to make our app more visually appealing. For simplicity I don’t Remember, I’m using SASS, so you might need to compile below code.

(CSS)

At this point we should have something like presented in the picture below. Also we’re done with HTML and CSS and from now on we’ll be working only with JavaScript.

(add image)

## Hands on Web Speech API

We’ve mentioned earlier Chrome supports SpeechRecognition with prefixed properties, therefore in first few lines we include the following code to feed the right objects to right browsers.

(var Speech Recognition = SpeechRecognition || webkitSpeechRecognition…)

Next we need to define a speech recognition instance to control the recognition in our app. This is done using the SpeechRecognition() constructor.

(var recognition = new SpeechRecognition())

https://www.smashingmagazine.com/2014/12/enhancing-ux-with-the-web-speech-api/  
<https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html#tts-section>  
<http://caniuse.com/#search=speech>  
<http://lists.w3.org/Archives/Public/public-webapps/2011OctDec/att-1696/speechapi.html>  
<https://www.adobe.com/devnet/archive/html5/articles/voice-to-drive-the-web-introduction-to-speech-api.html>  
<https://developer.mozilla.org/en-US/docs/Web/API/SpeechSynthesisUtterance>  
https://www.sitepoint.com/introducing-web-speech-api/  
<https://developers.google.com/web/updates/2013/01/Voice-Driven-Web-Apps-Introduction-to-the-Web-Speech-API>  
<https://developer.mozilla.org/en-US/docs/Web/API/SpeechRecognition>  
<https://developer.mozilla.org/en-US/docs/Web/API/SpeechGrammarList>  
https://developer.mozilla.org/en-US/docs/Web/API/Web\_Speech\_API