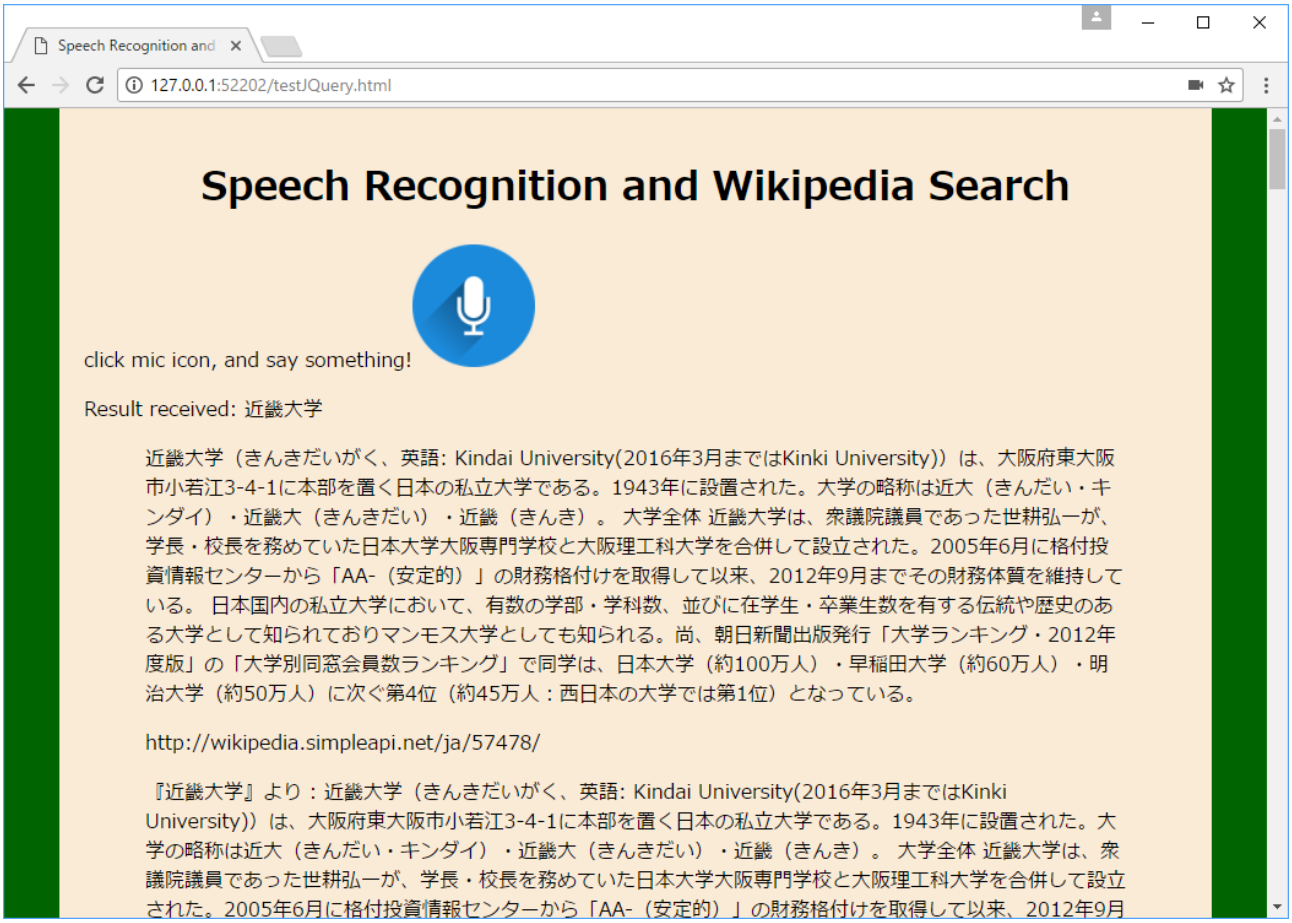
近畿大学工学部「公開講座」　　2017 年 5 月 20 日（土）

JavaScript プログラミング基礎講座

徐　丙鉄（情報学科）

制作する Web アプリ：音声入力 + Wikipedia 検索

音声でキーワードを入力し，Wikipedia を検索して，結果を表示する。



スケジュール

1．開発環境 Brackets のインストール
2．HTML と CSS の復習，CSS アニメーション
　＜休憩＞
3．JavaScript と jQuery
4．SimpleAPI：WikipediaAPI（Wikipedia 検索 API）
　＜休憩＞
5．Web Speech API の紹介
6．アプリの制作

学習サイト

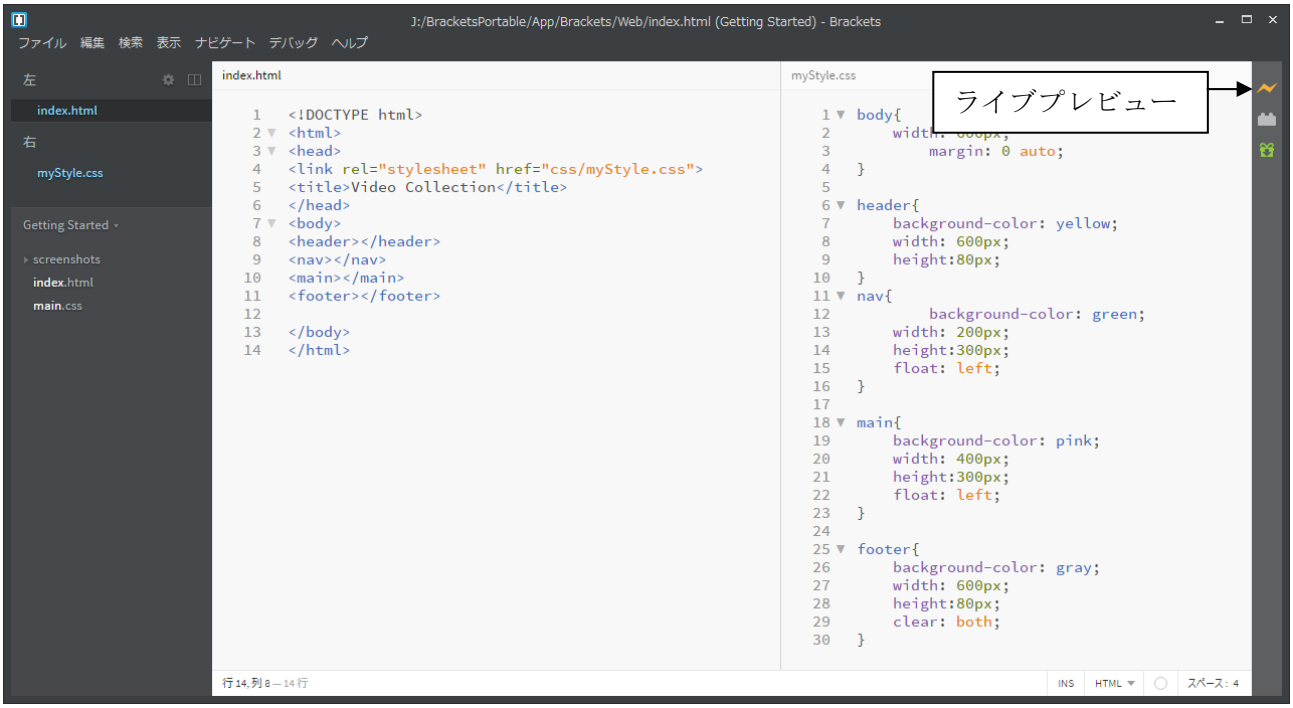
- ドットインストール　http://dotinstall.com/
- Schoo　https://schoo.jp/guest
- Skill Hub　http://skillhub.jp/

学習資料：以下のキーワードで検索して資料を取得する。

- HTML cheat sheet
- CSS cheat sheet
- JavaScript cheat sheet
- jQuery cheat sheet

１．Brackets：Web による，Web のための次世代エディタ

A modern, open source text editor that understands web design



表示→左右分割：左に html ファイル，右に css ファイル

便利機能

・Getting Started をクリックして，作業フォルダを選択すると，フォルダ内がリスト表示される。

・シンタックスハイライト：HTML，CSS，JavaScript，PHP，Ruby，Python，C，・・・

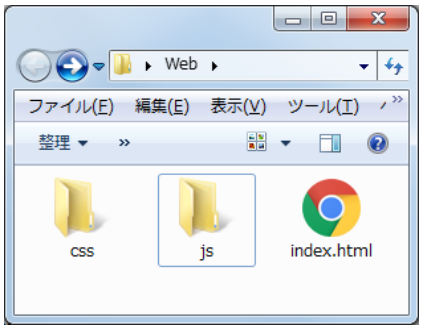
・コードヒント【Ctrl+Space】(HTML，CSS，JavaScript）を表示

・inline editor【Ctrl+E 】：CSS のクイック編集，Esc キーを押せば、HTML の編集に戻る
HTML の要素や id，class などを右クリックして「クイック編集」を選択又は【Ctrl+E 】で，直下にクイック編集ウインドウが開き CSS の編集ができる。

・ライブプレビュー

ライブプレビュー	【Ctrl+Alt+P】
ライブプレビューを強制的に再読込み	【Ctrl+Shift+R】
ライブプレビューハイライト	【Ctrl+Shift+C】

Brackets には Node.js が同梱されており，Web サーバーが起動し，現在編集状態となっている HTML ファイルを HTTP リクエストで Web ブラウザ（Chrome）で表示してくれる。



USB メモリに Brackets をインストールする

1. 「brackets portable」で検索
2. Download BracketsPortable [34.4MB download / 131MB installed]
3. ダウンロードした「BracketsPortable_Release_1.9_English.paf.exe」ダブルクリックする。
4. install 先として USB メモリを指定する。
5. USB メモリ内の「BracketsPortable」フォルダ内に，この演習用に「Web」フォルダを作る。
6. Web フォルダ内に css フォルダと js フォルダを作る。
7. Brackets を起動し，**Getting Started** をクリックし，Web フォルダを選択する。

２．HTML と CSS の復習，CSS アニメーション

WWW(World Wide Web)

インターネット上の Hyper Text で，連想記憶のようにリンクで情報を結びつけることで，Web 上の情報を構造化し提供するサービス。

HTML(Hyper Text Markup Language)

Web 文書を記述する言語で，Web 文書の論理構造をタグでマークアップして記述する。

例： <h1>**近畿大学工学部「公開講座」**</h1>

searchWikipedia.html

```
<!DOCTYPE html>
<html lang="ja">
<head>
<meta charset="UTF-8">
<title>Speech Recognition and Wikipedia Search</title>
</head>
<body>

<h1>Speech Recognition and Wikipedia Search</h1>
<p>click mic icon, and say something! <img src="mic.png" id="mic"></p>

<div id="output"></div>

<div id="wikipediaResults"></div>

</body>
</html>
```

CSS(Cascading Style Sheet)

Web 文書のスタイル（体裁，レイアウト，視覚表現）を指定する。

Cascading：スタイルの指定が上流から下流へ引き継がれる。

大域的スタイル　→　局所的なスタイル

外部 CSS ファイル → HTML の head のスタイル → HTML タグ内のスタイル

ブラウザ（user agent stylesheet）　→　Web ページの CSS　→　user style sheet

構文：selector { property: value; }

例： h1{ color: darkgreen; text-align: center; }

```
#wikipediaResults{
        width: 800px;   margin: 0 auto;
}

#mic{
        width: 100px;
        animation-duration: 2s;
        animation-iteration-count: infinite;
        animation-name: record;
}
```

searchWikipedia.html

```
<h1>Speech Recognition and Wikipedia Search</h1>
<p>click mic icon, and say something!
<img src="mic.png" id="mic"></p>
<div id="wikipediaResults"></div>
```

```
@keyframes record{
      0%{ opacity: 1; }
     50%{ opacity: 0.2; }
    100%{ opacity: 1; }
}
```

３．JavaScript と jQuery

JavaScript：ブラウザで実行可能なプロトタイプベースのオブジェクト指向のスクリプト型プログラム言語：識別子（変数名、関数名）には，英数字，$，_ 意外は使えない。

animation-name ×　　animationName ○

jQuery (write less, do more) : DOM 要素問合せ(query) のための JavaScript ライブラリ

John Resig @Mozilla Corporation　発表：2006 年 1 月

jQuery 関数

jQuery()，そのエイリアス **$**()

(1) CSS セレクタによるドキュメントの DOM ノード選択：セレクタ構文で指定

$('#mic')　id="mic" の要素の jQuery オブジェクト

$('button')　　button 要素の jQuery オブジェクト

注：DOM(Document Object Model):HTML ファイルを HTML 要素のツリー構造として記述する

(2) 要素の属性値を取得，変更

取得:　var imgSrc = $('img').**attr**('src');

変更:　$('img').**attr**({
　　src: '/images/hat.gif',
　　title: 'jQuery',
　　alt: 'jQuery Logo'
　});

(3) 要素の css を変更

$('#you').**css**('background-color','#ff0000')

id="you" を持つ要素の背景色を赤に

document.getElementById('you').style.backgroundColor = '#ff0000'

$('p').**css**('color','#ff0000')

p 要素全ての文字色を赤に

document.getElementsByTagName('p').style.color = '#ff0000'

(4) 要素内容の変更

$('#wikipediaResults').**text**('Hello');

id=" wikipediaResults" 要素の内容を，指定した text で置き換える。

$('#wikipediaResults').**html**('That is Wikipedia .');

id=" wikipediaResults" 要素の内容を，指定した HTML で置き換える。

$('#wikipediaResults').**append**('近畿大学工学部');

id=" wikipediaResults" 要素の内容の最後に，指定した内容を追加する。

(5) イベント処理

click, dblclick, mousedown, mouseup, mouseover, mouseout, mousemove

```
$('#mic').click(
    function() {
        alert('クリックしましたね');
  }
);
```

id="mic" の要素がクリックされたら，指定した処理を実行する。

４．SimpleAPI：WikipediaAPI（Wikipedia 検索 API）：http://wikipedia.simpleapi.net/

検索例

http://wikipedia.simpleapi.net/api?keyword=TCP/IP
http://wikipedia.simpleapi.net/api?keyword=TCP/IP&output=json
http://wikipedia.simpleapi.net/api?keyword=TCP/IP&output=json&callback=myFunction

SimpleAPI：WinkipediaAPI 入力仕様

API の URI：http://wikipedia.simpleapi.net/api

パラメータは，GET あるいは POST メソッドで指定する。入力文字コードは現在 UTF-8 のみ。出力は UTF-8 固定。

・keyword：キーワード。エイリアスとして、q でも指定可能。
・output：出力方式（xml, rss, json, html, javascript, php, tsv を指定可能。デフォルトは xml）
・callback：出力形式が JSONP 時の名称を指定可能
・lang：現在未実装。現在は日本語（ja）のみ。
・search：現在未実装につき 1 のみ。（0.その特定キーワードのみを取り出す、1.前方一致、2.後方一致、3.前後方一致、4.FULLTEXT）

SimpleAPI：WikipediaAPI 出力仕様

出力形式
・language：言語名
・id：Wikipedia 内の管理コード
・url：Wikipedia にリンクする際の URL
・title：キーワードのタイトル
・body：本文のダイジェスト
・length：本文の長さ
・redirect：別キーワードへのリダイレクトがあるかどうか。あれば 1、なければ 0。
・strict：そのキーワードと完全一致する場合には 1。完全一致しない場合は 0。
・datetime：更新日付

jQuery による Ajax 通信:
http://wikipedia.simpleapi.net/api?keyword=近畿大学&output=json&callback= logResults

---

```
var keyword = '近畿大学' ;
var wikipediaApiUrl = 'http://wikipedia.simpleapi.net/api' ;
wikipediaApiUrl = wikipediaApiUrl + '?keyword= ' + keyword + '&output=json&callback=logResults' ;
        $.ajax({
            type: 'GET',
            url: wikipediaApiUrl,
            dataType: 'jsonp',
            jsonpCallback: 'logResults'
        });
```

Response

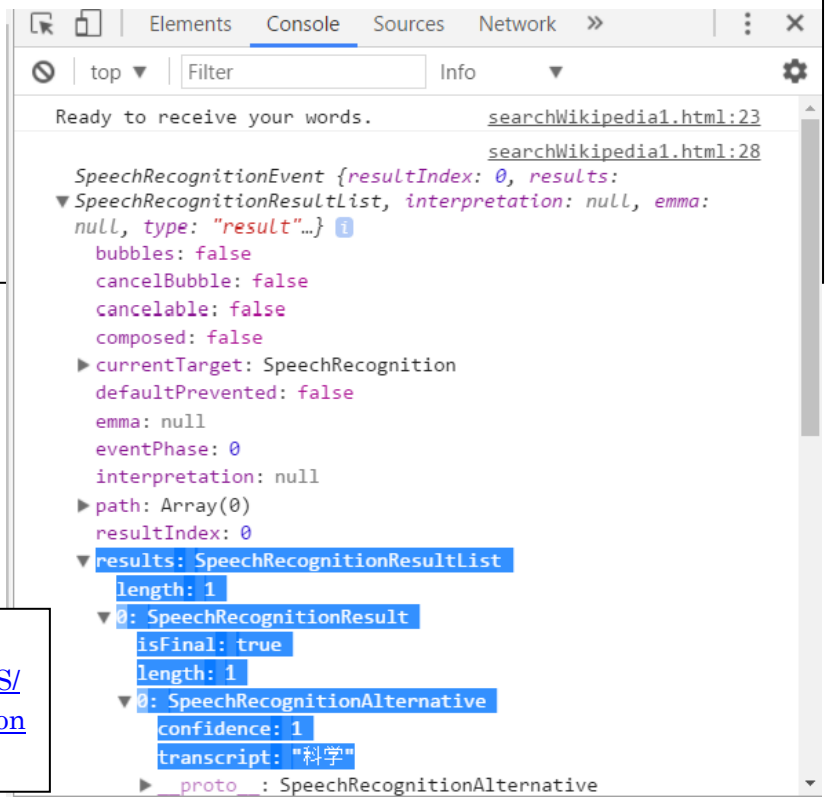
```
logResults(
    [{language: "ja", id: "57478",
      url: http://wikipedia.simpleapi.net/ja/57478/",
      title: "近畿大学",…},…]
)
```

Ajax: Asynchronous JavaScript + XML
XMLHttpRequest（HTTP 通信を行うための JavaScript 組み込みクラス）による非同期通信

５．Web Speech API

searchWikipedia.html

```
<!DOCTYPE html>
<html lang="ja">
<head>
<meta charset="UTF-8">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
<title>Speech Recognition and Wikipedia Search</title>
</head>
<body>
<h1>Speech Recognition and Wikipedia Search</h1>
<p>click mic icon, and say something! <img src="mic.png" id="mic"></p>

<div id="output"></div>
<div id="wikipediaResults"></div>

<script>
var recognition = new webkitSpeechRecognition();          //Chrome の場合接頭辞 webkit
recognition.lang = 'ja';

$('#mic').click(
        function() {
                recognition.start();
                console.log('Ready to receive your words.');
        }
);

recognition.onresult = function(event) {
        console.log(event);
}

</script>
</body>
</html>
```



音声認識結果

event.results[0][0].transcript;

参照

https://developer.mozilla.org/en-US/docs/Web/API/SpeechRecognition/onresult

Web Speech API

機能：音声認識（Asynchronous Speech Recognition）と音声合成（Text-to-Speech）

---

(1) Web Speech API Specification　https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html

(2) Web Speech API: Add Speech to your Website

https://docs.google.com/document/d/1vH9Zf86XNuqIwUFg5OJn56Q2qP3HeMEUoezx0NPQddY/edit

(3) ブラウザサポート状況　http://caniuse.com/#feat=speech-recognition

１．音声認識 Asynchronous Speech Recognition

SpeachRecognition　recognition service のコントローラー

Constructor

SpeechRecognition()　Chrome の場合接頭辞 webkit が必要

Attribute

lang = 'ja'	認識言語設定
maxAlternatives = 1	認識結果として返される候補の数。default は 1。
interimResults =false	認識処理の途中でも、結果を返すなら true。default は false。
continuous = false	認識単位ごとに、結果を返すなら　true。default は false。

Methods

start()　音声認識開始.

stop()　音声認識を終了し，結果を返す。

EventHandler

onresult　recognition service が結果を返したときに呼び出される。

onspeechstart

onspeechend

onnomatch

onerror

onstart

onend

６．アプリの制作

```
<!DOCTYPE html>
<html lang="ja">
<head>
<meta charset="UTF-8">
<title>Speech Recognition and Wikipedia Search</title>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
<style>
html{
    width: 100%;
    background-color: darkgreen;
}
```

```
body{
	width: 900px;
	margin: 0 auto;
	padding: 20px;
	background-color: antiquewhite;
}
#mic{
	width: 100px;
	animation-duration: 2s;
	animation-iteration-count: infinite;
	animation-name: none;
}

@keyframes record{
	  0%{ opacity: 1; }
	 50%{ opacity: 0.2; }
	100%{ opacity: 1; }
}

#wikipediaResults{
		width: 800px;
		margin: 0 auto;
}

h1{ text-align: center; }
</style>
</head>
<body>
<h1>Speech Recognition and Wikipedia Search</h1>
<p>click mic icon, and say something! <img src="mic.png" id="mic"> </p>

<div id="yourWords"></div>
<div id="wikipediaResults"></div>

<script>
//wikipedia 検索結果の callback 関数
function logResults(wikiJsons){
	$('#wikipediaResults').text('');		// wikipedia 検索結果表示領域のリセット

	$.each(wikiJsons, function( i, json){
		var body = json.body;
		var url = json.url;
		$('#wikipediaResults').append( '<p>' + body + '</p>' + url );
	}) ;
}

//音声認識
var recognition = new webkitSpeechRecognition();
	recognition.lang = 'ja';
```

```
    recognition.onresult = function(event) {
        var yourWords = event.results[0][0].transcript;
        $( '#yourWords' ).text( 'Result received: ' + yourWords);

        //wikipedia 検索
        var wikipediaApiUrl   = 'http://wikipedia.simpleapi.net/api';
        wikipediaApiUrl = wikipediaApiUrl + '?keyword=' + yourWords + '&output=json&callback=logResults';
        $.ajax({
            type: 'GET',
            url: wikipediaApiUrl,
            dataType: 'jsonp',
            jsonpCallback: 'logResults'
        });

        stopAnime();
    }

$('#mic').click(
    function() {
        recognition.start();
        startAnime();
        $('# yourWords ').text( 'Ready to receive your words. ' );
    }
);

function startAnime(){
    $('#mic').css( 'animation-name',    'record' );
}

function stopAnime(){
    $('#mic').css( 'animation-name',    'none' );
}
</script>
</body>
</html>
```

音声合成 Text-to-Speech

例

```
<script>
        var synth = window.speechSynthesis;
        var utterance1 = new SpeechSynthesisUtterance('こんにちわ');
        synth.speak( utterance1 );
</script>
```

```
$.ajax({
  url: 'http://www.ekidata.jp/api/l/11302.json',
  dataType: 'jsonp', // 追加
  type: "GET",
  success: function(res) {
    console.log(res);
  }
});
```

3. jquery.xdomainajax.js を利用する

jquery.xdomainajax.js という jQuery プラグインを用いることで、普通に ajax で他ドメインにアクセスできるようになります。

```
<html>
<head>
<script src="jquery-2.0.3.min.js"></script>
<script src="jquery.xdomainajax.js"></script> <!-- 追加 -->
<script>
$.ajax({
  url: 'http://www.ekidata.jp/api/l/11302.json',
  type: "GET",
  success: function(res) {
    console.log(res);
  }
});
```

```
</script>

</head>

<body>

</body>

</html>
```

これで取得できたデータを見てみます。

```
function jsonCallback(json){
   console.log(json);
}

$.ajax({
   url: "http://run.plnkr.co/plunks/v8xyYN64V4nqCshgjKms/data-2.json",
   dataType: "jsonp"
});


function logResults(json){
   console.log(json);
}

$.ajax({
   url: "https://api.github.com/users/jeresig",
   dataType: "jsonp",
   jsonpCallback: "logResults"
});
```

以下のように書くこともできます。

```
$.getJSON("https://api.github.com/users/jeresig?callback=?",function(json){
   console.log(json);
});
```