

JAVASCRIPT入門講座 2021

54代 HarrisonKawagoe(@hrsnnkwge_pro)



JAVASCRIPTは何?

- Webサイトやアプリの開発などに使われるスクリプト型言語
- サイトのHTML&CSSを動的に変更することが可能

- React.jsやVue.jsなどのWebフレームワークを勉強する為にはJavascriptの知識が必要
- JavaとJavaScriptは全く別物

講座内容

- 1.環境の構築
- 2.基本的な文法(型、配列、ループ、関数など)
- 3.Javascriptのよくある使い方
- 4.簡単なアプリを作ってみよう!

1. 環境構築

VSCodeを使うとコーディングが楽になるのでオススメです



Visual Studio Code interface showing a JavaScript file named `index.js` being edited. The left sidebar displays the **EXTENSIONS** view, listing installed and recommended extensions.

Installed Extensions:

- Beautify** 1.5.0: Beautify code in place for VS Code. HookyQR
- HTML Preview** 0.2.5: Provides ability to preview HTML doc... Thomas Haakon Townsend
- Jupyter** 2021.3.684299474: Jupyter notebook support, interactive... Microsoft
- Python** 2021.3.680753044: Linting, Debugging (multi-threaded, r... Microsoft
- Remote - WSL** 0.54.6: Open any folder in the Windows Subs... Microsoft
- Ruby** 0.28.1: Ruby language support and debuggin... Peng Lv
- VSCode Ruby** 0.28.0: Syntax highlighting, snippet, and langu... Stafford Brunk
- vscode-reveal** 4.0.3: Show markdown as revealsJs presentat...

Recommended Extensions:

- Docker** 1.11.0: Makes it easy to create, manage, and ... Microsoft [Install](#)
- Debugger for Chrome** 4.12.12: Debug your JavaScript code in the Ch... Microsoft [Install](#)
- Debugger for Firefox** 2.9.3: Debug your web application or brows... Firefox DevTools [Install](#)

The main editor area shows the code for `index.js`:

```
1 $(document).ready(function() {
2     $('#submit_form').submit(function() {
3         $('#res').text('Processing...');
4         $('#res').css('color', 'black');
5         $.ajax({
6             'url': $('#form#submit_form').attr('action'),
7             'type': 'POST',
8             'data': $('#form#submit_form').serialize(),
9             'dataType': 'json',
10            'success': function(response) {
11                if (response.result.includes('error') || response.result.includes('Time Limit Exceed!')) {
12                    $('#res').css('color', 'red');
13                } else {
14                    $('#res').css('color', 'green');
15                }
16                $('#res').html(response.result.replace(/\\r?\\n/g, '<br>'));
17                $('#time').text("Time: " + response.timeusage + "ms");
18                $('#memory').text("MemoryUsage: " + response.memoryusage + "KB");
19            },
20        });
21        return false;
22    });
23 });
24
```

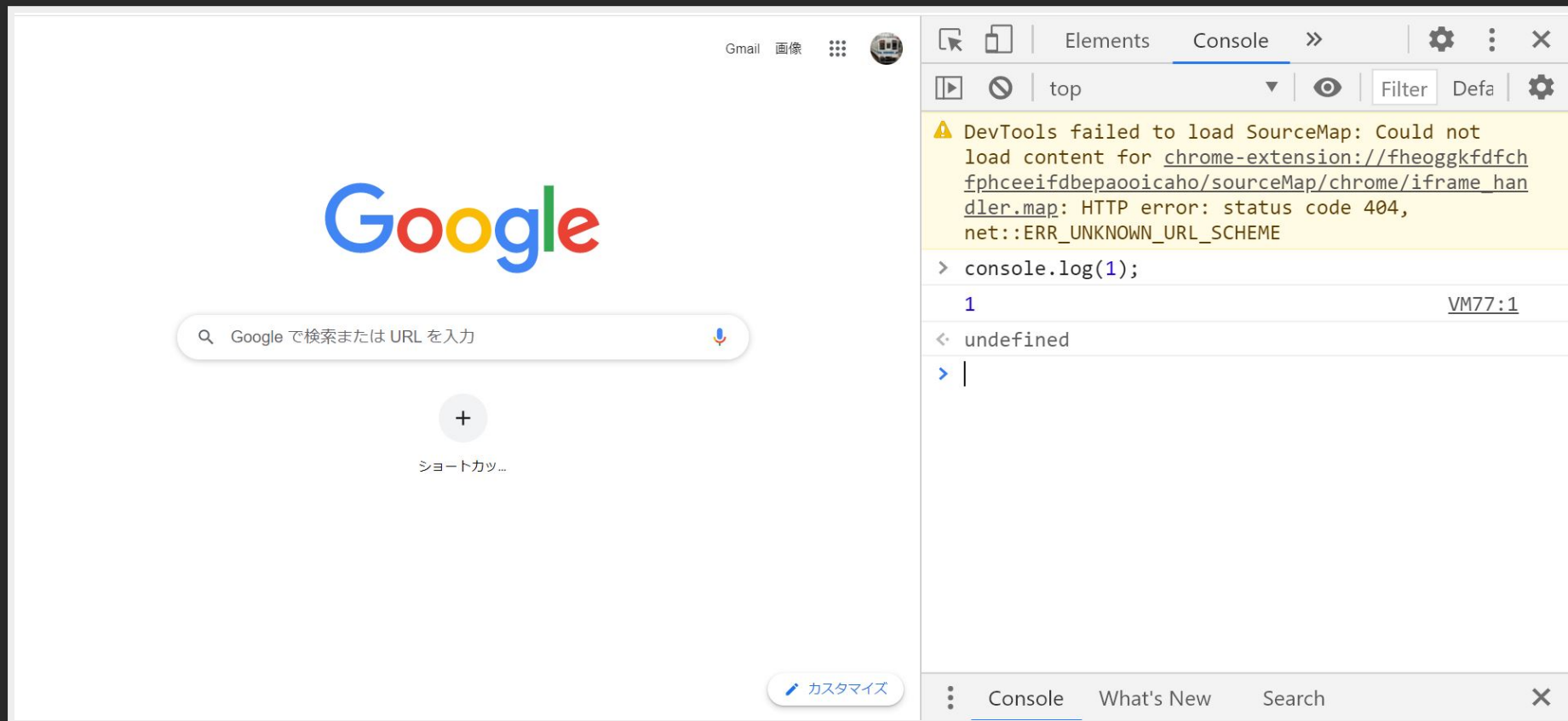
The status bar at the bottom indicates: `Ln 24, Col 1`, `Spaces: 4`, `UTF-8`, `CRLF`, `JavaScript`.

ダウンロード: <https://azure.microsoft.com/ja-jp/products/visual-studio-code/>

VSCODEが入ってない？

問題ありません!Chromeとメモ帳さえあればコーディングはできます！

新しいタブを開いて、F12ボタンを押してください



2.1 JAVASCRIPTの型、変数の宣言

- 文字列型 (string)、数値型 (number)、ブール型 (Boolean)、null型、undefined型などが存在する
- 変数に整数を代入した場合、整数として使うことはできるが、基本的にJavaScriptの数値型は浮動小数点数として表現される(C/C++のdoubleに相当)

Example :

```
let a = "mis.w"; //文字列型  
var b = 33; //数值型(整数)  
const c = 3.14; //数值型(実数)  
let d = true; //Boolean型
```

letとvarは変数、constは定数

```
let a = 1  
a = 2  
var b = 2  
b = 3  
//既に宣言された変数に新しい値を代入することが可能
```

constは再代入が許されないので、下記のようなコードを書くとエラーが発生する:

```
const a = 1  
a = 2
```

varは変数の再宣言が可能

```
var a = 2  
var a = 3
```

この書き方はあんまり良くないので、変数を使う際は
letを使うことをオススメします

2.2 算術演算子

```
let a = 1 + 2; //加算  
let b = 3 - 2; //減算  
let c = 1 * 2; //乗算  
let d = 7 / 3; //除算  
let e = 7 % 3; //剰余  
let f = 2 ** 3; //累乗
```

2.3 代入演算子

```
a += 4; // 加算代入, a = a + 4に相当  
a++; // インクリメント, a = a + 1に相当  
a -= 4; // 減算代入, a = a - 4に相当  
a--; // デクリメント, a = a - 1に相当  
a *= 4; // 乗算代入, a = a * 4に相当  
a /= 4; // 除算代入, a = a /= 4に相当
```


2.4 比較演算子&論理演算子

- 比較演算の結果はBoolean型(trueかfalse)として返ってくる
- 論理演算は条件の結合などに使われる

```
let a = 2 === 2; //等価
let b = 2 !== 2; //不等
let c = 3 > 2; //大なり
let d = 3 < 2; //小なり
let e = 2 >= 2; //以下なり
let f = 2 <= 3; //以上なり
let g = a & d; //AND演算
let h = a | d; //OR演算
```

「==」と「===」の違い

```
> 1 == 1
< true
> 1 == "1"
< true
> 1 === 1
< true
> 1 === "1"
< false
```

「===」はより厳密に比較してくれるので、こちらを使用することがオススメ

2.5.1 条件分岐(IF...ELSE...)

```
let a = 3;
if(a>3) {
  ...
}else if(a==3) {
  ...
}else if(a<3&&a>1) {
  ...
}else{
  ...
// () 内の条件は全て偽であればこのブロック内のコードが実行される
}
```

2.5.2 条件分岐(SWITCH)

if...else...と違って、case文に書く条件は値でなければならない

```
let a = 2;
switch(a){
  case 3:
    ...
  case 2:
    ...
    break; //breakしないと、これより下のcase文とdefault文の中身も実行
  case 1:
    ...
  default:
    ...
}
```

2.6 配列

1~100の数値をそれぞれ違う変数に保存する場合:

```
let a1 = 1;  
let a2 = 2;  
...  
let a100 = 100;
```

↑コードは読みづらいし、変数も使いづらい

配列の宣言:

```
let a = [1,2,...,100]; // 大量なデータを代入する際はループなどを使いましょう
```

要素の取得(index値は0~N-1):

```
a[0] // 最初の要素  
a[5] // 6番目の要素
```

要素の追加:

```
array.unshift(1); //1を先頭に追加  
array.push(1); //1を末尾に追加  
array.splice(1,0,4,5) //指定した位置に要素を追加、この場合2番目の要素の前に
```

要素の削除:

```
array.shift(); //最初の要素を削除  
array.splice(1,1); //2番目の要素を削除  
array.splice(1,2); //2番目から始まる要素を2個削除  
array.pop(); //最後の要素を削除
```

2.7.1 ループ (FOR...)

似たような処理を複数回行う場合はループ文を利用したほうが便利

```
let array=[];  
for(let a=1; a<=100; a+=1){  
    //初期値はa=1, aが100に到達するまでaの値が1ずつ増える  
    array.push(a);  
}
```


2.7.2 ループ(Foreach)

配列の要素を順番に取り出す時はこっちが便利

```
let array=[3,4,4,5,1,2];  
array.forEach(function(element) {  
    console.log(element);  
});
```

2.7.3 ループ(WHILE)

```
let a = 1;
while(a<10){
  a+=3;
}
console.log(a);
```

条件によっては無限ループになったりするので注意が必要

2.8 関数

```
let a=[3,4,4,5,1,2];
let result = 0;
a.forEach(function(element) {
    result+=element;
});
console.log(result);
let b=[6,2,3,1];
result = 0;
b.forEach(function(element) {
    result+=element;
});
console.log(result);
```

同じことを2回やってるけどコードを減らすことはできないかな?

関数を宣言すると、コードの使い回しが可能になる

```
//関数の宣言
function getSum(array){
    let result = 0;
    array.forEach(function(element){
        result+=element;
    });
    return result;//値を返す
}
```

```
let a=[3,4,4,5,1,2];
console.log(getSum(a));
let b=[6,2,3,1];
console.log(getSum(b));
```



下記ファイルを作成してください:

index.html

```
<script src="script.js"></script>
<input id="num">
<button type="number" id="click">Click ME!</button>
<div id="result">result</div>
```

script.js

```
//ここにJSのコードを記述
```

| | | | |
|--|-------------------|-------------------|------|
|  index.html | 4/10/2021 1:23 AM | Chrome HTML Do... | 1 KB |
|  script.js | 4/10/2021 1:22 AM | JavaScript ファイル | 0 KB |

3.1 イベントの追加

ボタンをクリックした後の挙動などを定義することができる

```
window.onload = function() {  
    //イベントはページの読み込みが終わってから設定する  
    let button = document.getElementById('click');//要素の取得  
    button.onclick = function onClick(){  
        alert('Click');  
    }  
}
```

HTML要素の取得する際に使われる関数:

- `getElementById()`
- `getElementsByClassName()`
- `querySelector()`
- `querySelectorAll()`

3.2 HTML要素の変更

内容の編集

```
let result = document.getElementById('result');  
result.innerHTML = "0";//内容の編集
```

CSS要素の変更

```
let result = document.getElementById('result');  
result.style.color = "red";//resultを赤くする
```


3.3 HTML要素の削除と挿入

テキストの挿入:

```
let result = document.getElementById('result');  
let elementText = "testText";  
result.append(elementText);
```

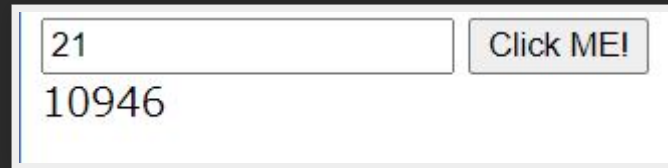
HTML要素の挿入:

```
let result = document.getElementById('result');  
let elementHTML = "<button>Button2</button>";  
result.insertAdjacentHTML('afterend', elementHTML); //afterend以外は
```

要素の削除:

```
let result = document.getElementById('result');  
result.remove();
```

4. 例題: フィットボナッチ数列のN項目を計算するWEBアプリを作ってみよう!



A screenshot of a web application interface. It features a text input field containing the number '21'. To the right of the input field is a button labeled 'Click ME!'. Below the input field, the result '10946' is displayed.

ヒント:

- 配列をaと定義した場合、n番目の要素
は: $a[n]=a[n-1]+a[n-2]$
- $a[0] = 0, a[1] = 1, a[2] = 1$
- 関数やループなどを活用すると上記の漸化式を実装できる

書き換える部分(script.js):

```
window.onload = function() {  
    let button = document.getElementById('click');  
    button.onclick = function onButtonClick() {  
        let n = parseInt(document.getElementById('num').value); //  
        //Write Code Here  
    }  
}
```

制限時間:5min



解答例1:

```
window.onload = function() {  
    let button = document.getElementById('click');  
    button.onclick = function onButtonClick(){  
        let n = parseInt(document.getElementById('num').value);  
        let array = [0,1,1];  
        let index = 2;  
        while(index<=n){  
            array[index]=array[index-1]+array[index-2];  
            index++;  
        }  
        result.innerHTML = array[n];  
    }  
}
```

解答例2:

```
window.onload = function() {  
  
    let button = document.getElementById('click');  
    button.onclick = function onClick(){  
        let n = parseInt(document.getElementById('num').value);  
        function fib(n){  
            if(n===0) return 0;  
            else if(n===1||n===2) return 1;  
            return fib(n-1)+fib(n-2);  
        }  
        result.innerHTML = fib(n);  
    }  
}
```

興味ある人は:

- JavaScript Primer:<https://jsprimer.net/>
- Progate:<https://prog-8.com/>
- ドットインストール:<https://dotinstall.com/>
- プロ研資料置き場:<https://misw.github.io/>

ご清聴ありがとうございました!