プログラミング演習(JavaScript)① 基本構造、配列、関数、探索、整列

■配布ファイル

1. HTML ファイルの場合

この演習では、主に HTML ファイルに JavaScript のコードを書いていく。コードは「Script」タグの中にコードを書いていく。Script タグは head タグでも body タグの中であれば動くが、多くの場合 body の終了タグの直前に記述する。

js0-1.htm

```
001 <!DOCTYPE html>
002 <html lang = ”ja”>
003   <head>
004     <meta charset = ”utf-8”>
005     <title>ここにタイトルを記述する</title>
006   </head>
007   <body>
008     <script>
009       //ここにプログラムを記述する
010     </script>
011   </body>
012 </html>
```

2. 順次配列

**上から順に実行している。文字列を開業するとき、改行コード¥n を入れる。js1-1-2 のように、文字列を指定する「’ (シングルコーテーション)」の中に
を入れても改行されない。**

js1-1.htm

```
    <略>
008     <script>
009       var a = prompt(’名前を入力してください’,’’);
010       var b = ’Hello, world!¥n’+ ’こんにちは、’+ a +’さん’;
011       alert(b);
012       document.write(b);
013     </script>
    <略>
```

js1-1-2.htm

```
    <略>
008     <script>
```

```
009         var a = prompt('名前を入力してください','');
010         var b = 'Hello, world!<br>'+ 'こんにちは、'+ a +'さん';
011         alert(b);
012         document.write(b);
013       </script>
014   <略>
```

2. 選択構造

点数と合格点を入力して、点数が合格点より低かったら、不合格、そうでなかったら合格を表示させる。switch 文を使って選択構造を作ることもある。(変数が0だったら不合格、1だったら合格のような文)

js1-2.htm

```
009         var a = Number(prompt('点数の入力',''));
010         var b = Number(prompt('合格点の入力',''));
011         if(a < b){
012           alert('残念！不合格です');
013         }else{
014           alert('おめでとう！合格です');
015         }
```

3. 反復構造

I で1から指定した数まで1ずつ増やし、kにk+i を代入して和を求めている。 sで文字列を結合させて、式を作っている。

js1-3.htm

```
009         var n = Number(prompt('1から N までの和を求める¥n'+'N を入力',''));
010         var s = 0;
011         var k = 0;
012         for(var i =1 ; i <  n ; i++ ){
013           s= s + i+'+';
014           k= k + i;
015         }
016           s= s + n;
017           k= k + n;
018           alert(s+'='+k);
```

なお、JavaScript ファイルの拡張子は.js とし、JavaScript ファイルに書くコードは script タグでは囲まない。

2. 配列

複数の値を含素の名前で管理するのではなく、一つの名前で管理する。例えば、月の名前を管理するときに、a=’Jan’ ， b=’Feb’ ， c=’mar’ とするのではなく、a[0]=’Jan’ ， a[1]=’Feb’ ，a[2]=’Mar’ と変数に添え字(インデックス)を付けて値(要素)を管理をする。 1つの添え字で指定する配列を1次元配列と言い、2つの添え字で指定する配列を1次元配列と言う。

●1次元配列

a[0]	a[1]	a[2]
Jan	Feb	Mar

●1次元配列の宣言例

```
001 var a = [’Jan’,’Feb’,’Mar’];
```

```
001 var a = [];
002 a[0]=’Jan’;
003 a[0]=’Feb ’;
004 a[0]=’ Mar ’;
```

```
001 var a = [];
002 a.push(’Jan’);
003 a.push(’Feb’);
004 a.push(’Mar’);
```

●2次元配列

a[0,0]	a[0,1]	a[0, 2]
a[1,0]	a[1,1]	a[2, 2]
a[2,0]	a[2,1]	a[2, 2]

Jan	Feb	Mar
Apr	May	Jun
Jul	Aug	Sep

●2次元配列の宣言例

```
001 var a = [ [’Jan’,’Feb’,’Mar’] , [’Apr’,’May’,’Jun’] , [’Jul’,’Aug’,’Sep’] ];
```

例　1, 2, 3を横と縦に並べる

js2-1.htm

```
009   var a = [1,2,3];
010       for(var i =0 ; i < 3 ; i++ ){
011         document.write(a[i]+' ');
012       }
013       document.write('<br><br>');
014       for(var i =0 ; i < 3 ; i++ ){
015         document.write(a[i]+'<br>');
016       }
```

結果

```
1 2 3

1
2
3
```

●2次元配列

GREEN TEA MANDARA(http://kadode-ooigawa.jp/green_tea_mandara/) を参考にして、表を作る

表　GREEN TEA MANDARA

		j			
		0	1	2	3
i	0	I	Ho	Ri	Wa
	1	Ro	He	Nu	Ka
	2	Ha	To	Ru	Yo
	3	Ni	Chi	Wo	Ta

js2-2.htm

```
009     var a =
010      [
011        ['I ','Ho','Ri','Wa'],
012        ['Ro','He','Nu','Ka'],
013        ['Ha','To','Ru','Yo'],
014        ['Ni','Chi','Wo','Ta'],
015      ];
016    for(var i=0; i<4; i++){
017        for(var j=0; j<4 ; j++){
018          document.write(a[i][j] +' ');
019        }
020      document.write('<br>');
021    }
```

■結果

I Ho Ri Wa
Ro He Nu Ka
Ha To Ru Yo
Ni Chi Wo Ta

例 タグを入れて表を作る

js2-2-2.htm

```
009      var a =
010      [
011        ['I',' Ho','Ri','Wa'],
012        ['Ro',' He','Nu','Ka'],
013        ['Ha',' To','Ru','Yo'],
014        ['Ni','Chi','Wo','Ta'],
015      ];
016  document.write('<table border="1">');
017    for(var i=0; i<4; i++){
018      document.write('<tr>');
019        for(var j=0; j<4 ; j++){
020         document.write('<td width="30" height="30" align="center">' + a[i][j] + '</td>');
021        }
022      document.write('</tr>');
```

```
023     }
024      document.write('</table>');
```

■結果

I	Ho	Ri	Wa
Ro	He	Nu	Ka
Ha	To	Ru	Yo
Ni	Chi	Wo	Ta

●関数

ここでは、入力した火入れと蒸しの2つの数を引数にして、二次元配列から、茶葉の種類、お湯の温度、抽出時間をさんしょうして、配列変数 kekka で返す関数を作る。

		mushi				
		0	1	2	3	4
hiire	0	I	Ho	Ri	Wa	70
	1	Ro	He	Nu	Ka	70
	2	Ha	To	Ru	Yo	90
	3	Ni	Chi	Wo	Ta	90
	4	90	90	60	60	

お湯の温度

抽出時間

js3-1.htm

```
009 function mandara($hiire,$mushi){
010   var a=
011   [
012    ['I','Ho','Ri','Wa','70'],
013    ['Ro','He','Nu','Ka','70'],
014    ['Ha','To','Ru','Yo','90'],
015    ['Ni','Chi','Wo','Ta','90'],
016    ['90','90','60','60',''],
017   ];
```

```
018     var chaba = a[$hiire][$mushi];
019     var ondo = a[$hiire][4];
020     var jikan = a[4][$mushi];
021     return[chaba,ondo,jikan];
022 };
023 var hiire = Number(prompt('火入れ(0 から 3 までの数字を入力してください)' , '' ));
024 var mushi = Number(prompt('蒸し(0 から 3 までの数字を入力してください)' , '' ));
025 var kekka = mandara(hiire,mushi);
026 document.write('<br><br>あなたが選んだお茶のタイプは「'+ kekka[0] + '」' );
027 document.write('<br>お湯の温度は' + kekka[1] +'℃、抽出時間は'+  kekka[2] + '秒です。');
```

■ 火入れ1、蒸し2を入力した結果

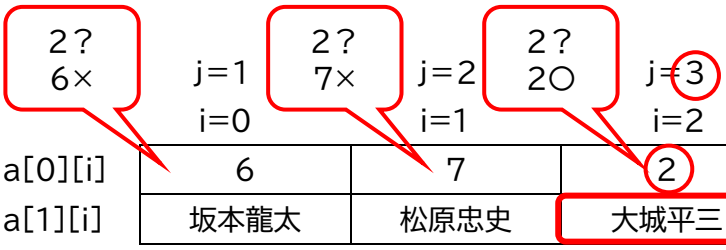
あなたが選んだお茶のタイプは「Nu」
お湯の温度は70℃、抽出時間は60秒です。

4. 探索

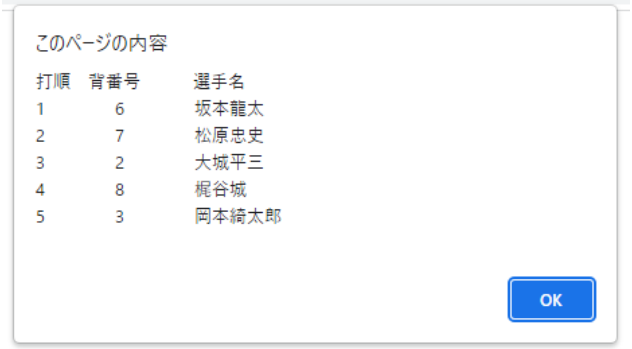
背番号を検索して、選手名と何番目に位置しているかを表示させる

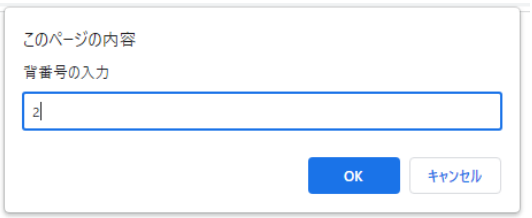
	i=0	i=1	i=2	i=3	i=4
a[0][i]	6	7	2	8	3
a[1][i]	坂本龍太	松原忠史	大城平三	梶谷城	岡本綺太郎

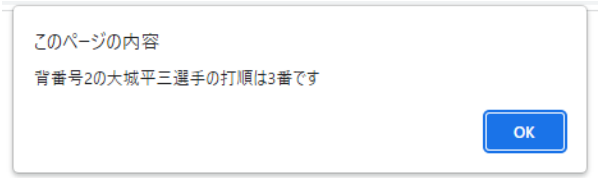
a[0][i]を順に見ていき、指定した値と同じだったら、番号や a[1][i]の値を返す。
検索する値を2とすると、i=0 から1ずつ増やして、a[0][i]のと同じになったら、jの値を返す



	i=0	i=1	i=2	i=3	i=4
a[0][i]	6	7	2	8	3
a[1][i]	坂本龍太	松原忠史	大城平三	梶谷城	岡本綺太郎







配列変数の定義

●順次探索

js4-1.htm

```
009     var a =
010       [
011         ['6','7','2','8','3'],
012         ['坂本龍太','松原忠史','大城平三','梶谷城','岡本綺太郎']
013       ];
014     var n = a[0].length;
015     var msg = '打順  背番号    選手名¥n';
016     for (var i = 0 ; i < n ; i++) {
017      j=i+1;
018        msg = msg + j + '      ' + a[0][i] + '      ' + a[1][i] + '¥n';
019      }
020     alert(msg);
021     var s = Number(prompt('背番号の入力',''));
022     for(i=0; i<n; i++){
023       if(a[0][i] == s){
024         j=i+1;
025          alert('背番号'+s +  'の'+ a[1][i] +'選手の打順は' + j + '番です');
026         break;
027         }
028       }
```

配列変数の定義

配列変数の内容の表示

検索結果の表示

参考　選手成績(2021 年 4 月 23 日現在)

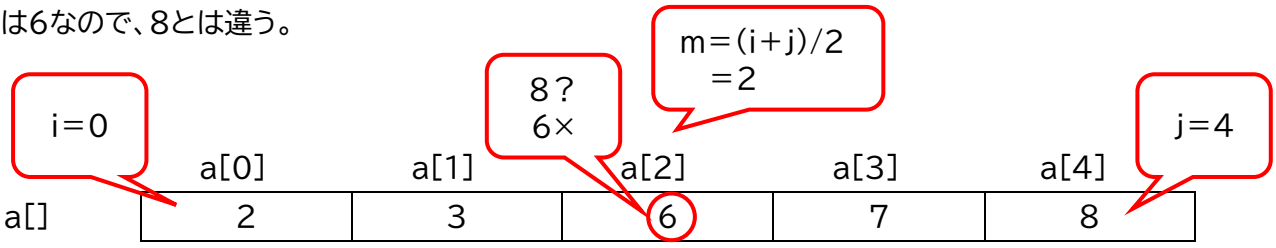
https://www.giants.jp/smartphone/G/report/batter.html

●2分探索

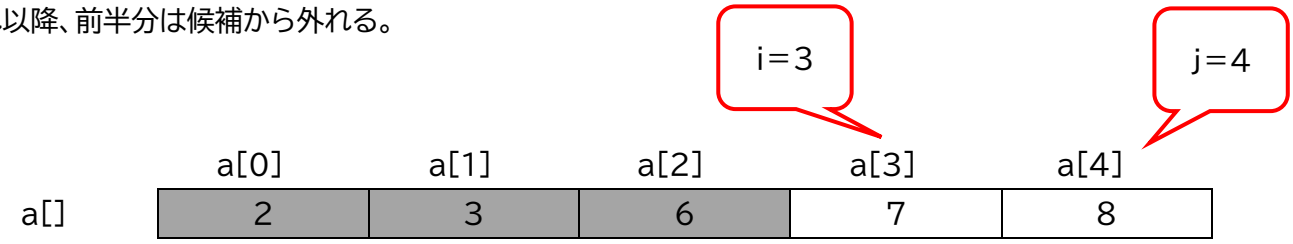
整列をした状態で使える探索方法。この方法は、候補を半分ずつに減らしていける。
1列目だけ見る

	a[0]	a[1]	a[2]	a[3]	a[4]
a[]	2	3	6	7	8

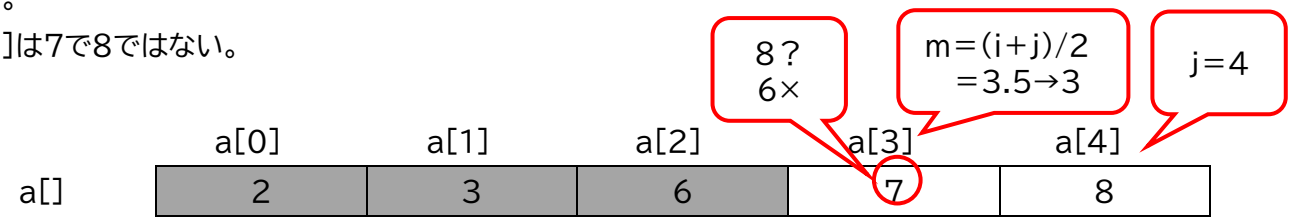
8を例にして考える
範囲を初めの0と後ろの4にする。(i=0、j=4) すると中央の位置は (i+j) / 2 = 2
a[2]は6なので、8とは違う。



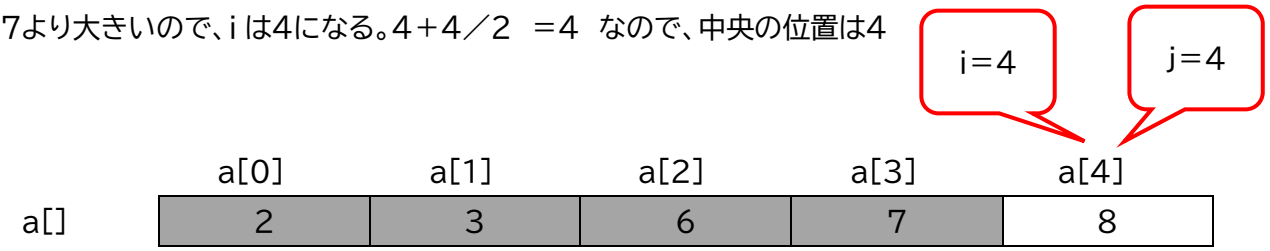
8は6より大きいので、範囲を真ん中の位置より後ろ半分にする。
i を中央の場所の2＋1の3とする。(もし、指定した値が小さかったら、jを中央の場所の2―1の1とする。)
これ以降、前半分は候補から外れる。



3と4の中央の場所は(3＋4)＝3．5で小数になるが、Math.floor に入れているので切り捨てられて3になる。
a[3]は7で8ではない。



8は7より大きいので、i は4になる。4＋4／2 ＝4 なので、中央の位置は4



a[4]＝8なので、8が探索できた。 何番目か表示させるため、jun にm＋1を代入して、打順を表示させた

js4-2.htm

```
009     var a =
010       [
011         ['2','3','6','7','8'],
012         ['大城平三','岡本綺太郎','坂本龍太','松原忠史','梶谷城']
013       ];
014       var n = a[0].length;
015       var msg = '順番 背番号    選手名¥n';
016       for (var i = 0 ; i < n ; i++) {
017         jun =i+1;
018         msg = msg + jun + '     ' + a[0][i] + '     ' + a[1][i] + '¥n';
019         }
020      alert(msg);
021      var s = Number(prompt('背番号を入力してください',''));
022      i = 0;
023      var j = n - 1;
024      while( i <= j ){
025         var m = Math.floor(( i + j ) / 2 );
026         if(a[0][m] == s){
027          jun=m+1
028          alert('背番号' + s + 'の' + a[1][m] + 'は' + jun + '番です。');
029          break;
030       }
031       if( a[0][m] > s ){
032         j = m - 1;
033       }else{
034         i = m + 1;
035       }
036     }
```

●整列　交換法(バブルソート)①

整列のアルゴルズムは交換法以外にもあり、交換法は理解がしやすいが処理に時間がかかるアルゴリズムである。下記のリンクを参考にするとよいだろう。

※整列のアルゴリズムの比較
Visualization and Comparison of Sorting Algorithms
https://youtu.be/ZZuD6iUe3Pc

ここでは、一番後ろから隣り合ったものを比べて、前の位置の値が大きかったら入れ替える。比べる場所を一つ前にして同じようなことを繰り返していくと、一番手前に、一番前に一番小さな値が来る。
次に一番後ろから前から2番目まで暗い返すと、整列される。

a[0]	a[1]	a[2]	a[3]	a[4]
6	7	2	8	3

1番後ろから隣り合った値を比べ、手前が大きかったら入れ替える。ここでは、手前の8の方が大きいので、入れ替えている。

6	7	2	3	8

次の隣り合った値を比べて、同じ処理をしていく。ここでは、手前の2の方が小さいので入れ替えない。

6	7	2	3	8

7の方が大きいので入れ替える。

6	2	7	3	8

6の方が大きいので入れ替える。1番手前の2が確定する。

2	6	7	3	8

1番後ろに戻って、2番目まで比べていく(比較する回数を1回減らす)。3の方が小さいので入れ替えない。

2	6	7	3	8

7の方が大きいので入れ替える。

2	6	3	7	8

6の方が大きいので入れ替える。2番目の3が確定する

2	3	6	7	8

1番後ろに戻って、3番目まで比べていく。7の方が小さいので入れ替えない。

2	3	6	7	8

6の方が小さいので入れ替えない。3番目が確定する。

2	3	6	7	8

1番後ろに戻って、4番目まで比べていく。7の方が小さいので入れ替えない。4番目、5番目が確定する。

2	3	6	7	8

js5-1

```
009       var a = ['6','7','2','8','3'];
010       var n = a.length;
011       var msg = '整列前¥n 通し番号　背番号¥n';
012       for (i = 0 ; i < n ; i++) {
013        var jun = i+1
014        msg = msg + jun + '      ' + a[i] + '¥n';
015       }
016     alert(msg);
017      for (i = 0 ; i < n-1 ; i++) {
018        for(var j = n-2 ; j > i-1 ; j--) {
019           if(a[j] > a[j+1]){
020            var temp = a[j];
021            a[j] = a[j+1];
022            a[j+1] = temp;
023          }
024        }
025      }
026     var msg = '整列後¥n 通し番号　背番号¥n';
027       for (i = 0 ; i < n ; i++) {
028        var jun = i+1
029        msg = msg + jun + '      ' + a[i] + '¥n';
030       }
031     alert(msg);
```

●整列　交換法①‘

　比べる方向を変える。1番手前から隣り合った値を比べ、手前が大きかったら入れ替える。ここでは、手前の6の方が小さいので、入れ替えない。

6	7	2	3	8

　次の隣り合った値を比べて、同じ処理をしていく。ここでは、手前の7の方が大きいので入れ替える。それを繰り返していく。

6	2	7	3	8
6	2	3	7	8
6	2	3	7	8

1番後ろの位置の値に1番大きな値がくる。1番初めに戻って、処理を繰り返す。

2	6	3	7	8
2	3	6	7	8
2	3	6	7	8
2	3	6	7	8

2	3	6	7	8
2	3	6	7	8

js5-1-2

```
009       var a = ['6','7','2','8','3'];
010       var n = a.length;
011     var msg = '整列前¥n 通し番号　背番号¥n';
012       for (i = 0 ; i < n ; i++) {
013        var jun = i+1
014        msg = msg + jun + '      ' + a[i] + '¥n';
015       }
016     alert(msg);
017      for (i = 0 ; i < n-1 ; i++) {
018        for(var j = i ; j < n-1-i ; j++) {
019           if(a[j] > a[j+1]){
020            var temp = a[j];
021            a[j] = a[j+1];
022            a[j+1] = temp;
023          }
024        }
025      }
026     var msg = '整列後¥n 番号　背番号¥n';
027       for (i = 0 ; i < n ; i++) {
028        var jun = i+1
029        msg = msg + jun + '      ' + a[i] + '¥n';
030       }
031     alert(msg);
```