Javascript 入門

コレクション

鄭祥飛@2021/05/19

コレクションの種類

- インデックス付きコレクション（例：配列）
- キー付きコレクション　（例：Map）

【一致】
一つ変数に複数データが保存できる。

【相違】：各要素の保存方法が差異があります。
インデクス付き：順番がある。
キー付き：順番がなし

```
// 配列
let arr = ["cds","ddd","dfsafd","adfd","fdsad"];
for(let data of arr) {
    console.log(data);
}

// Map
let test = new Map();
sayings.set('dog', 'woof');
sayings.set('cat', 'meow');
sayings.set('elephant', 'toot');

for (let [key, value] of sayings) {
  console.log(key + ' goes ' + value);
}
```

配列

配列の作成

以下の３つ方法があります

```
let arr1=['a','b','c'];
let arr2= new Array('a','b','c');
let arr3= Array('a','b','c');
```

お勧め： let arr = [];

配列の長さの指定

```
let arr1=[];
arr1.length = 100;

let arr2= new Array(100);
let arr3= Array(100);
```

配列長さの変更

- 配列の長さ： 最終の要素のIndex＋ 1
- 配列の長さを変更すると、配列の内容を変更できる。

```
let cats = ['Dusty', 'Misty', 'Twiggy'];
cats.length = 2;
console.log(cats);
```

上記、'Dusty', 'Misty', だけ出力される

配列の循環処理

1. for
2. for ... of
3. for ... in
4. forEach

for ... in

indexとユーザー定義プロパティ名前で循環

```
let cats = ['Dusty', 'Misty', 'Twiggy']
cats['name']="Mars";
for(let c in cats) {
    console.log(c);
}
```

for ... of

配列中の各値を循環で処理する

```
let cats = ['Dusty', 'Misty', 'Twiggy']
cats['name']="Mars";
for(let c of cats) {
    console.log(c);
}
```

forEach

循環処理で、「break」と「continue」は無効です。
forEach中のパラメータは関数です。

```
['a','b','c'].forEach(element => {
    console.log(element);
});
```

循環処理を中止したい場合、どうする?

配列のメソッド

配列変数の後ろに処理関数を追加できる。

- forEach()
- concat() / join()
- push() / pop() / shift() / unshift()
- slice() / splice()
- reverse() / sort ()
- indexOf() / lastIndexOf() / filter()
- map() / every() / some() / reduce()

concat()

1. 2つ配列を結合し、新しい配列を返す
2. パラメータ： 配列

```
let arr1 = ['a','b'];
let arr2 = ['c','d'];
let arr3 = arr1.concat(arr2);
```

arr3 : ['a','b','c','d']

join()

1. すべての要素を文字列に結合します。
2. パラメータ： 結合文字

```
let myArray = new Array('Wind', 'Rain', 'Fire')
let list = myArray.join(' - ')
```

文字列のsplit()関数の逆操作。

push()

1. 配列の最後に一つ要素を追加する
2. パラメータ：任意値

```
let myArray = new Array('1', '2')
myArray.push('3')
```

pop()

1. 配列の最後の要素を取り除き
2. パラメータ： なし

```
let myArray = new Array('1', '2', '3')
let last = myArray.pop()
// myArray は ["1", "2"] に、last は "3" になる
```

unshift()

1. 配列の先頭に一つ以上の要素を追加する、その新しい配列の長さを返します。
2. パラメータ：任意値

```
let myArray = new Array('1', '2', '3')
myArray.unshift('4', '5')
// myArray は ["4", "5", "1", "2", "3"] になる
```

shift()

1. 配列から最初の要素を取り除き、その要素を返します
2. パラメータ： なし

```
let myArray = new Array('1', '2', '3')
let first = myArray.shift()
// myArray は ["2", "3"]に、first は "1" になる
```

slice(start_index, upto_index)

配列の一部を抽出し、新しい配列を返します。
start_indexからupto_indexまでの要素を抽出して返す。
upto_indexの要素が含めていない

```
let myArray = new Array('a', 'b', 'c', 'd', 'e')
myArray = myArray.slice(1, 4)
// インデックス 1 から始め、インデックス 3 まですべての要素を
// 展開して、[ "b", "c", "d"] を返す
```

splice(index, count_to_remove,addElment ,...)

indexからcount_to_removeまでの要素を取り除き、
そして、「addElment,...」の内容を挿入にする

```
let myArray = new Array('1', '2', '3', '4', '5')
myArray.splice(1, 3, 'a', 'b', 'c', 'd')
// myArray は ["1", "a", "b", "c", "d", "5"] になる
// このコードは、インデックス 1 の要素（"2" のあった場所）から始まり、
// 3 つの要素を削除して、そこに後続のすべての要素を挿入します。
```

reverse()

配列を反転させる

```
let myArray = new Array('1', '2', '3')
myArray.reverse()
// 配列要素が入れ替えられ、myArray = ["3", "2", "1"] になる
```

sort((a,b)=>{})

```
let myArray = new Array('Wind', 'Rain', 'Fire')
myArray.sort((a,b)=>{
  if (a[a.length - 1] < b[b.length - 1]) return -1;
  if (a[a.length - 1] > b[b.length - 1]) return 1;
  if (a[a.length - 1] == b[b.length - 1]) return 0;
});
```

- a[a.length - 1] が b[b.length - 1] より小さいとされた場合、 -1 を返します
- a[a.length - 1] が b[b.length - 1] より大きいとされた場合、 1 を返します
- a[a.length - 1] と b[b.length - 1] が等値と見なされる場合、 0 を返します

indexOf(searchObj,formIndex)

- 配列中に「searchObj」という要素が存在するかどうか検索する。
- 最初に検索されたもののIndexを返す
- formIndex:必須ではない
- 見つかったない場合、「－１」を返す

lastIndexOf(searchObj,formIndex)

- 配列の後ろから「searchObj」という要素が存在するかどうか検索する。
- 最初に検索されたもののIndexを返す
- formIndex:必須ではない
- 見つかったない場合、「－１」を返す

map((item)=>{})

配列中の各要素を処理してから、返す

```
let a1 = ['a', 'b', 'c']
let a2 = a1.map(function(item) { return item.toUpperCase() })
console.log(a2) // ['A', 'B', 'C'] がログに出力される
```

filter((item)=>{})

検索

```
let a1 = ['a', 10, 'b', 20, 'c', 30]
let a2 = a1.filter((item)=>{
  return typeof item === 'number';
})
console.log(a2)  // [10, 20, 30] がログに出力される
```

every((item)=>{})

配列中にすべての要素を条件を満足する場合、Trueを返す

```
let a1 = [1, 2, 3];
console.log(a1.every(
  item => {
    return typeof item === 'number';
  }
));
```

some((item)=>{})

配列中に一つの要素を条件を満足する場合、Trueを返す

```
let a1 = [1, 2, 3];
console.log(a1.every(
  item => {
    return typeof item === 'number';
  }
));
```

reduce((accumulator,current)=>{})

各要素の累計処理

```
let a = [10,20,30];
let total = a.reduce((accumulator,current)=>{
  return accumulator - current;
});
console.log(total);
```

多次元配列

1. 配列に配列がある。

```
let arr = [[1,2,3,2],['a','b','c'],['one','two','three']];
```

2. 要素の取得

   arr[2][1]: 'two';

宿題

１.二次元の配列（５＊８）の作成する。配列の各要素はランダムで生成する。各要素の値は１−１００子配列の各要素は降り順で表示親配列の各要素は子配列のトータル値を合計して、昇順で表示する

２．連続数字の配列に中に、漏れの数字を検索する。例:[0,1,2,3,4,6,7,8,10]。漏れ：５、９

３．指定の配列に、前半は奇数、後半は偶数、奇数の部分は降り順で表示する、偶数の部分は昇順で表示する

Map

Mapの作成

```
let myMap = new Map();
```

Mapにデータを設定

```
let myMap = new Map();
myMap.set("wang","111");
myMap.set("zhang","111");
myMap.set("li","122");
myMap.set("wang","122");
```

Keyは唯一に存在する

Mapのサイズ

```
myMap.size;
```

Mapからデータ取得

```
myMap.get("wang");
```

Mapからデータ削除

```
myMap.set("wang",null);
myMap.delete("wang");
```

MapにKeyが存在するかどうか

```
myMap.has("wang");
```

Mapすべての要素を削除

```
myMap.clear();
```

Mapの循環処理

```
let myMap = new Map();
for (let [key, value] of myMap) {
  console.log(key + ' goes ' + value);
}
```

Set

SetとArrayの比較

- Setに複数のデータを管理する
- Set中の要素は一意です。重複できないです

```
let mySet = new Set();
mySet.add(1);
mySet.add('some text');
mySet.add('foo');
mySet.add(1);
mySet.size; // 3

mySet.has(1); // true
mySet.delete('foo');
for (let item of mySet) {
  console.log(item);
}
mySet.clear();
```

まとめ

プログラム　＝　データ　＋　ロジック

1. データは変数に保存する
2. データの種類はコレクション,非コレクション とそのたがあります.
3. コレクション： Array,Map,Set
4. 非コレクション： text, number,data ,boolean,
5. そのた： json,object,function
6. ロジック： if, for, for...in ,for...of, while, do...while,switch,break,continue,try...catch,throw

宿題

以下はa-zのモールスエンコードです

```
".-","-...","-.-.","-..",".","..-.","--.","....","..",".---","-.-",
".-..","--","-.","---",".--.","--.-",".-.","...","-","..-","...-",
".--","-..-","-.--","--.."
```

任意の複数の英語単語をモールスコードに変換して、結果が相違の個数を計算する。例：英語単語は下記の場合：

"gin", "zen", "gig", "msg"、それぞれのモールスは下記です

```
"gin" -> "--...-."
"zen" -> "--...-."
"gig" -> "--..--."
"msg" -> "--..--."
```

そうすると、結果相違の個数は２です。

宿題

0–20の数字をランダムで生成し、重複の数字を除外してから各値の合計値を計算する