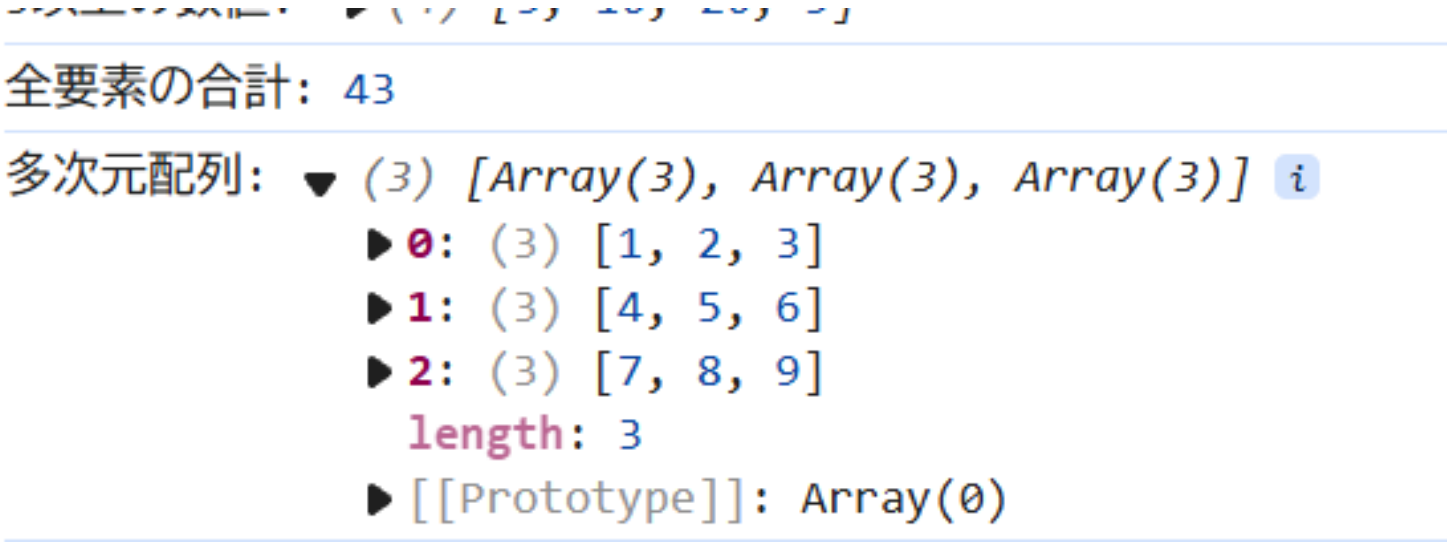
JSの基礎を学習してみよう2



この教材でできること

- JAVAscript(JS)について理解できる

  どんな教材？

  jsで計算をさせたり、配列でデータを代入したりできるよ

目次

①JSで計算をしてみよう

②1次元配列を使ってみよう

③多次元配列を使ってみよう

目次

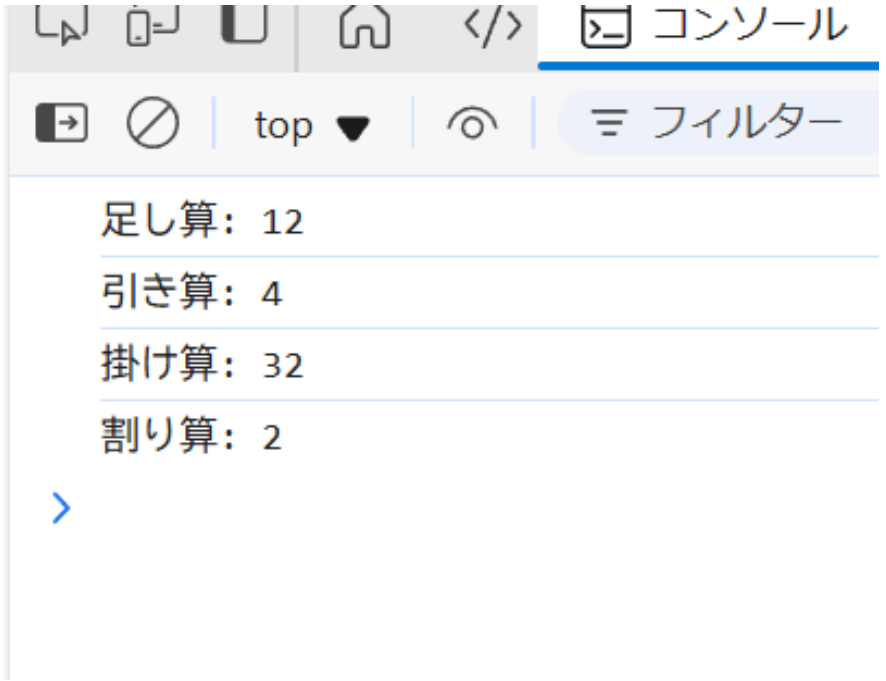
①JSで計算をしてみよう

②1次元配列を使ってみよう

③多次元配列を使ってみよう

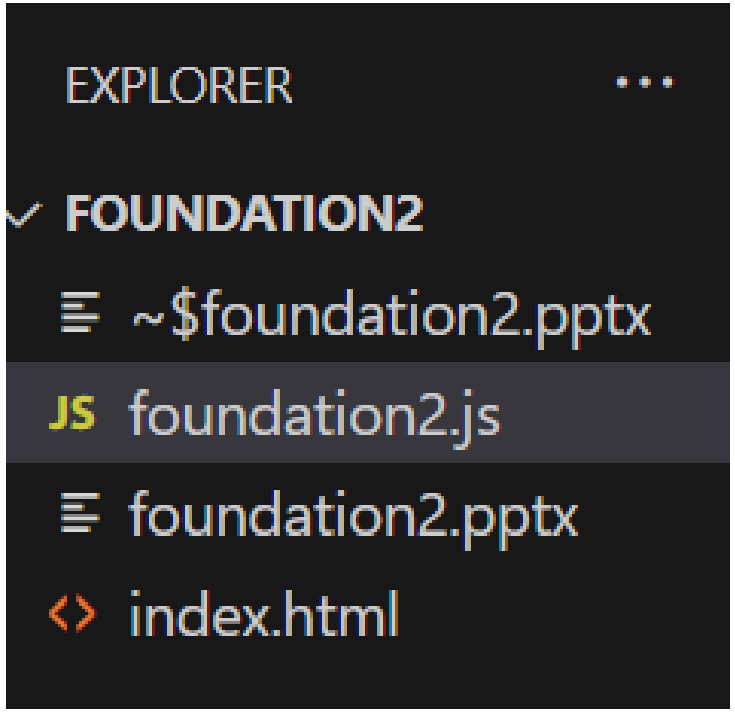
①JSで計算をしてみよう

このページでは計算結果がconsoleで表示できるよ



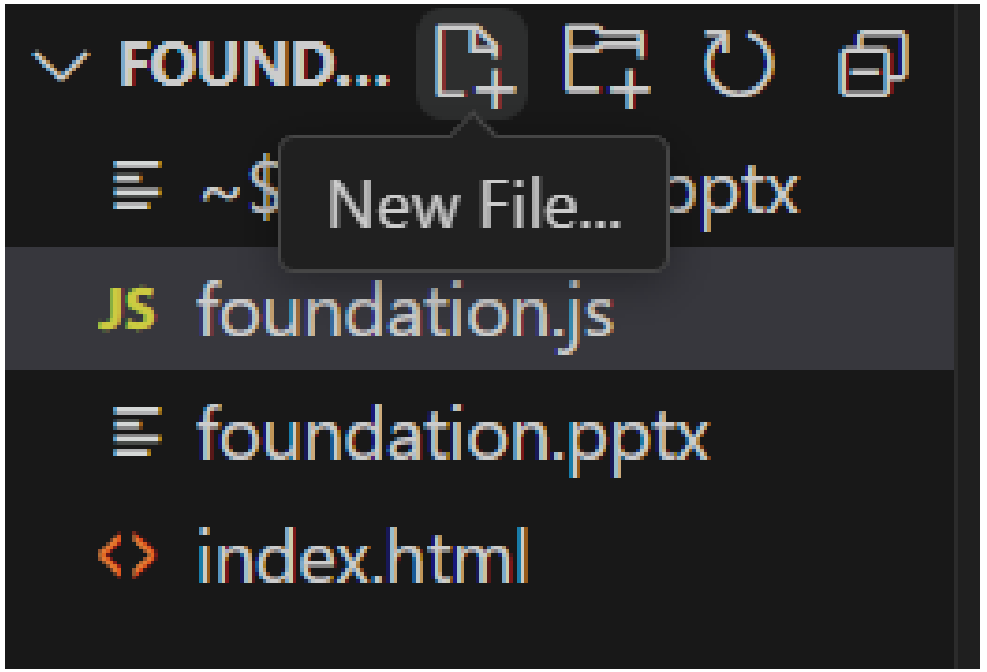
①JSで計算をしてみよう

まずはVscodeを開いてみよう



①JSで計算をしてみよう

htmlとjsのコードを作ってみよう



質問

htmlのコードとjsのコードを連携させるようなプログラミングを書いてみよう

※考えてから次のスライドに進んでみよう！

答え

htmlファイルに以下の内容を書いたらいいよ

index.html > script

```
1    <meta charset="UTF-8">
2    <script src="foundation2.js"></script>
```

①JSで計算をしてみよう

promptで2つの数字を入力できるようにしよう

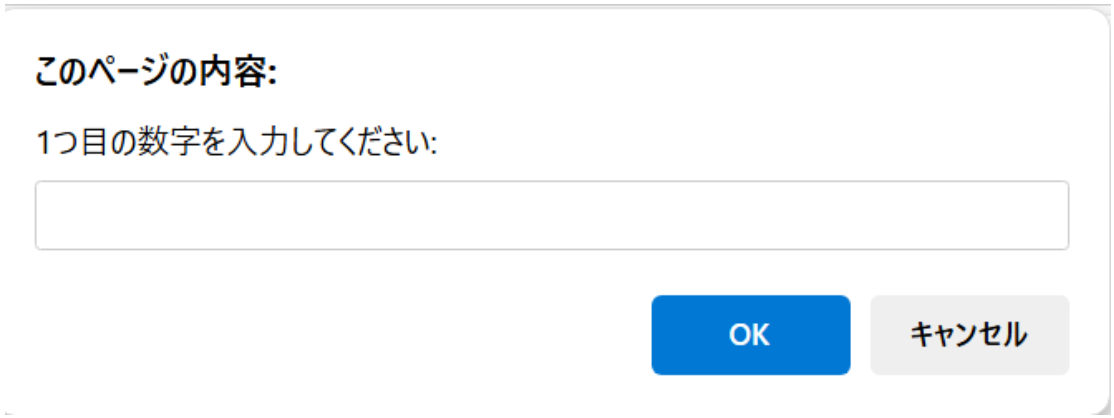
これが四則演算の元となる数字になるよ

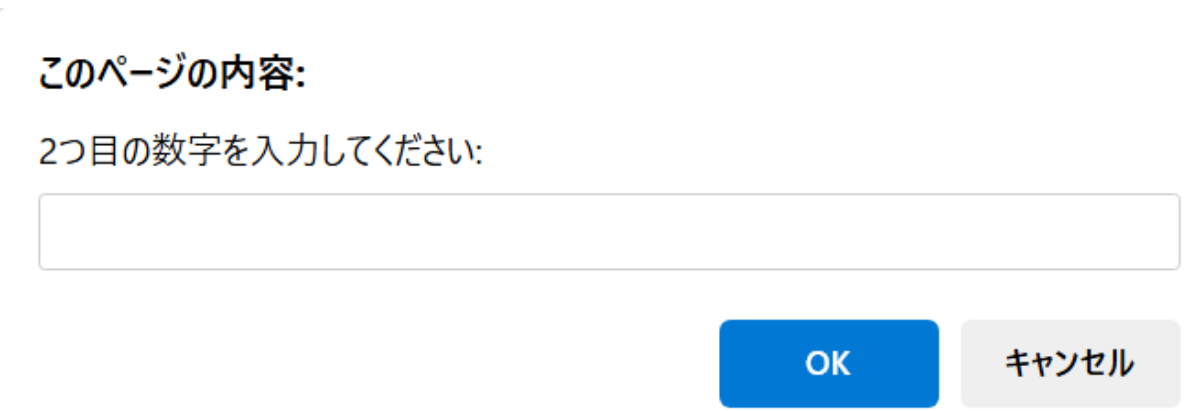
foundation2.js > ...

```
1    const num1 = prompt("1つ目の数字を入力してください:");
2    const num2 = prompt("2つ目の数字を入力してください:");
3
```

①JSで計算をしてみよう

htmlを開いて2つ入力できるようになったかな？





①JSで計算をしてみよう

次に入力した数字をJSの変数に対応するようにするよ

JS foundation2.js > ...

```
1    const num1 = prompt("1つ目の数字を入力してください:");
2    const num2 = prompt("2つ目の数字を入力してください:");
3
4    const number1 = parseFloat(num1);
5    const number2 = parseFloat(num2);
```

①JSで計算をしてみよう

console.logで四則演算の結果を表示するよ

割り算だけ少し書き方が違うよ

```
const number1 = parseFloat(num1);
const number2 = parseFloat(num2);

console.log("足し算: " + (number1 + number2));
console.log("引き算: " + (number1 - number2));
console.log("掛け算: " + (number1 * number2));
console.log("割り算: " + (number2 !== 0 ? (number1 / number2) : "0で割ることはできません"));
```

説明

(判定？処理1:処理2):判定を行ってその判定があっていたら処理1を行い、判定が間違っていたら処理2を行う

例)

(number2 !== 0？(number1 / number2) : "0で割ることはできません")

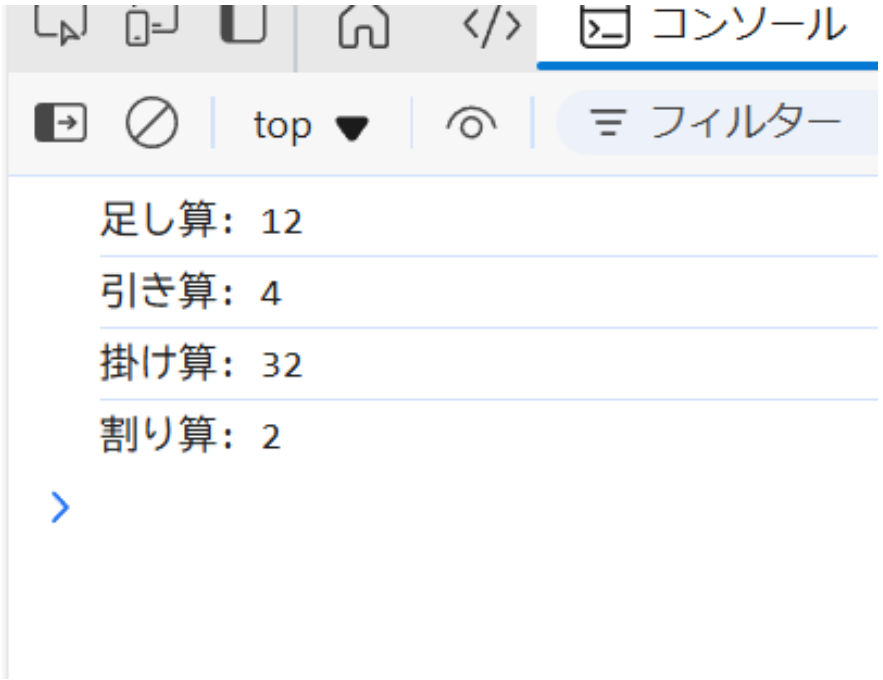
①number2が**0ではない時**number1/number2を行う。

②number2が**0の時**「0で割ることはできません」と表示

割り算では0で割ることができないため、このような書き方になるよ

①JSで計算をしてみよう

四則演算の結果がうまくいくか確認してみよう



質問

次に例外処理について考えよう！

ここでうまく実行できない(バグが起きそうな)場合は何が考えられるかな？

※考えてから次のスライドに進んでみよう！

答え

今回はこの２つの例外処理について考えていくよ

- 入力されたものが数字以外のものの場合
- 入力されなかった場合

①JSで計算をしてみよう

まずは数字以外の入力をした場合の処理を書こう！

```
const num1 = prompt("1つ目の数字を入力してください:");
const num2 = prompt("2つ目の数字を入力してください:");


const number1 = parseFloat(num1);
const number2 = parseFloat(num2);

if (!isNaN(number1) && !isNaN(number2)) { // 有効な数字か確認
    console.log("足し算: " + (number1 + number2));
    console.log("引き算: " + (number1 - number2));
    console.log("掛け算: " + (number1 * number2));
    console.log("割り算: " + (number2 !== 0 ? (number1 / number2) : "0で割ることはできません"));
} else {
    console.log("有効な数字を2つ入力してください。");
}

```

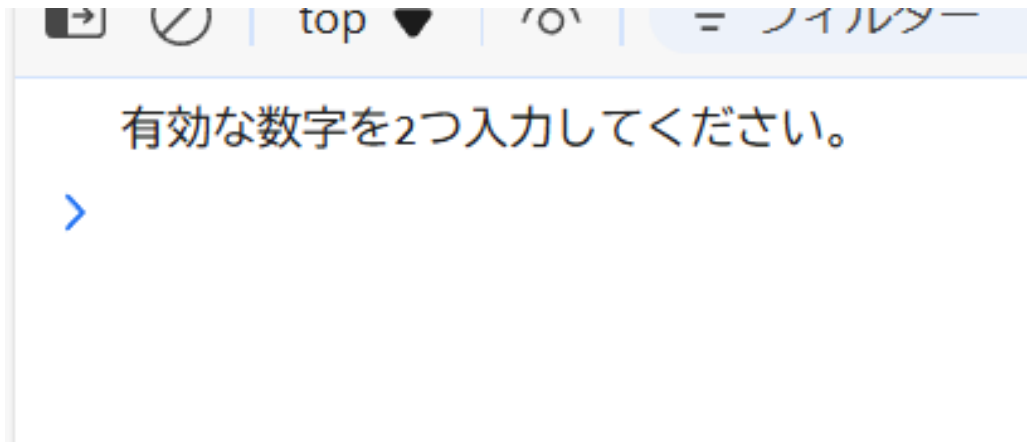
①JSで計算をしてみよう

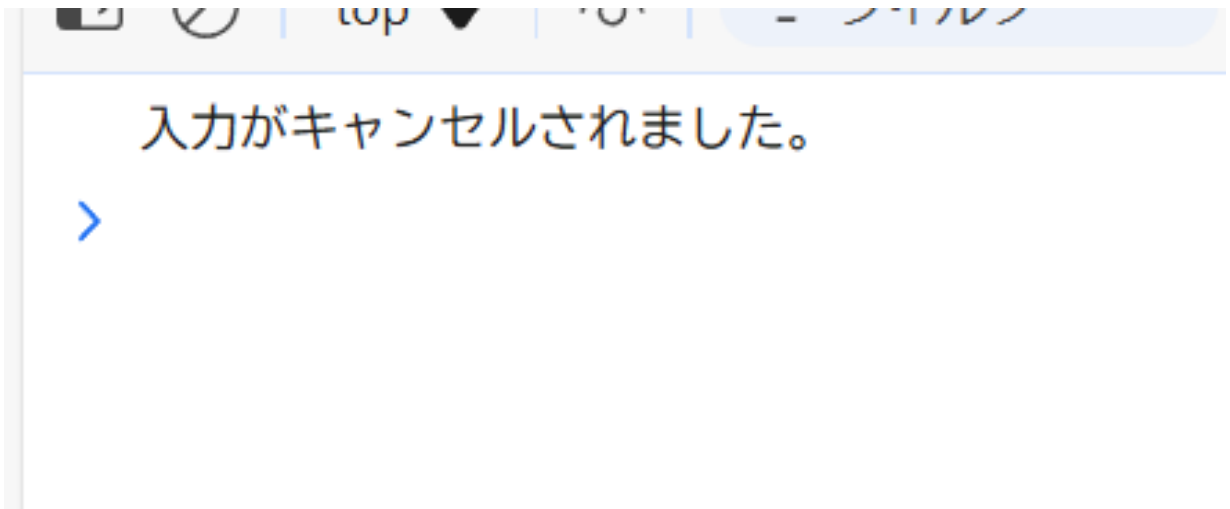
次に入力せずにOKボタンを押された場合の処理を書こう

```
const num1 = prompt("1つ目の数字を入力してください:");
const num2 = prompt("2つ目の数字を入力してください:");

if (num1 !== null && num2 !== null) {

    const number1 = parseFloat(num1);
    const number2 = parseFloat(num2);

    if (!isNaN(number1) && !isNaN(number2)) { // 有効な数字か確認
        console.log("足し算: " + (number1 + number2));
        console.log("引き算: " + (number1 - number2));
        console.log("掛け算: " + (number1 * number2));
        console.log("割り算: " + (number2 !== 0 ? (number1 / number2) : "0で割ることはできません"));
    } else {
        console.log("有効な数字を2つ入力してください。");
    }
} else {
    console.log("入力がキャンセルされました。");
}

```

①JSで計算をしてみよう

実際に2つの例外処理がうまくいくか確認してみよう





目次

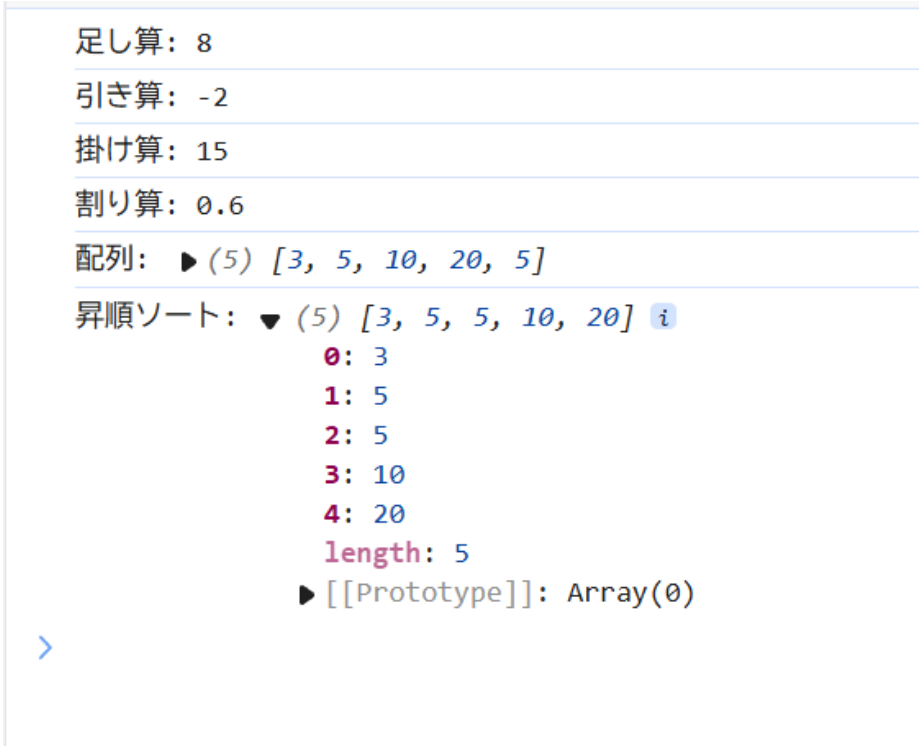
①JSで計算をしてみよう

②1次元配列を使ってみよう

③多次元配列を使ってみよう

②1次元配列を使ってみよう

このページでは配列を使っていろんなことを学習していくよ



②1次元配列を使ってみよう

配列とは同じような種類のデータを格納するデータ構造

例）

weights = [24, 40, 33]

体重が24, 40,33の人が3人いるような例だよ

②1次元配列を使ってみよう

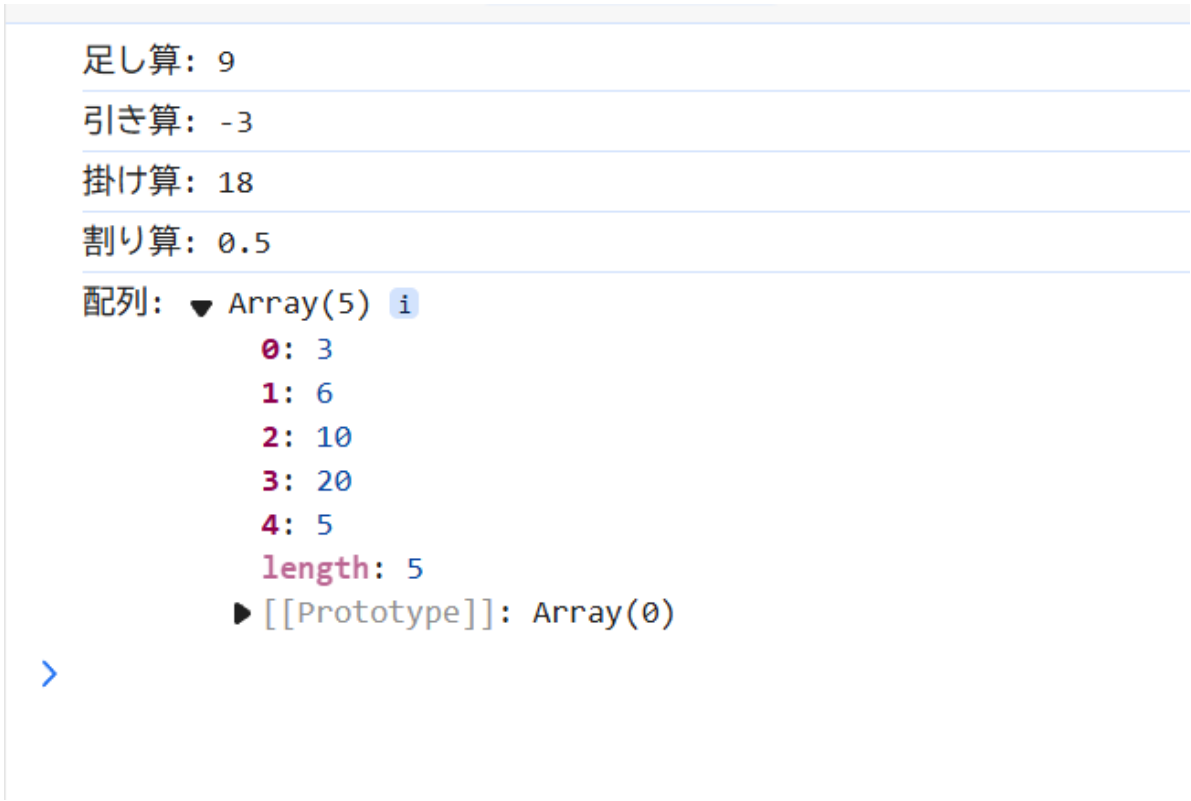
配列を定義しよう

最初の2つは入力した値で、3つ以降は決められた数字にしよう

```
11      // 四則演算の出力
12      console.log("足し算: " + (number1 + number2));
13      console.log("引き算: " + (number1 - number2));
14      console.log("掛け算: " + (number1 * number2));
15      console.log("割り算: " + (number2 !== 0 ? (number1 / number2) : "0で割ることはできません"));
16
17      // 配列を使った計算や操作
18      const numbers = [number1, number2, 10, 20, 5];
19      console.log("配列:", numbers);
```

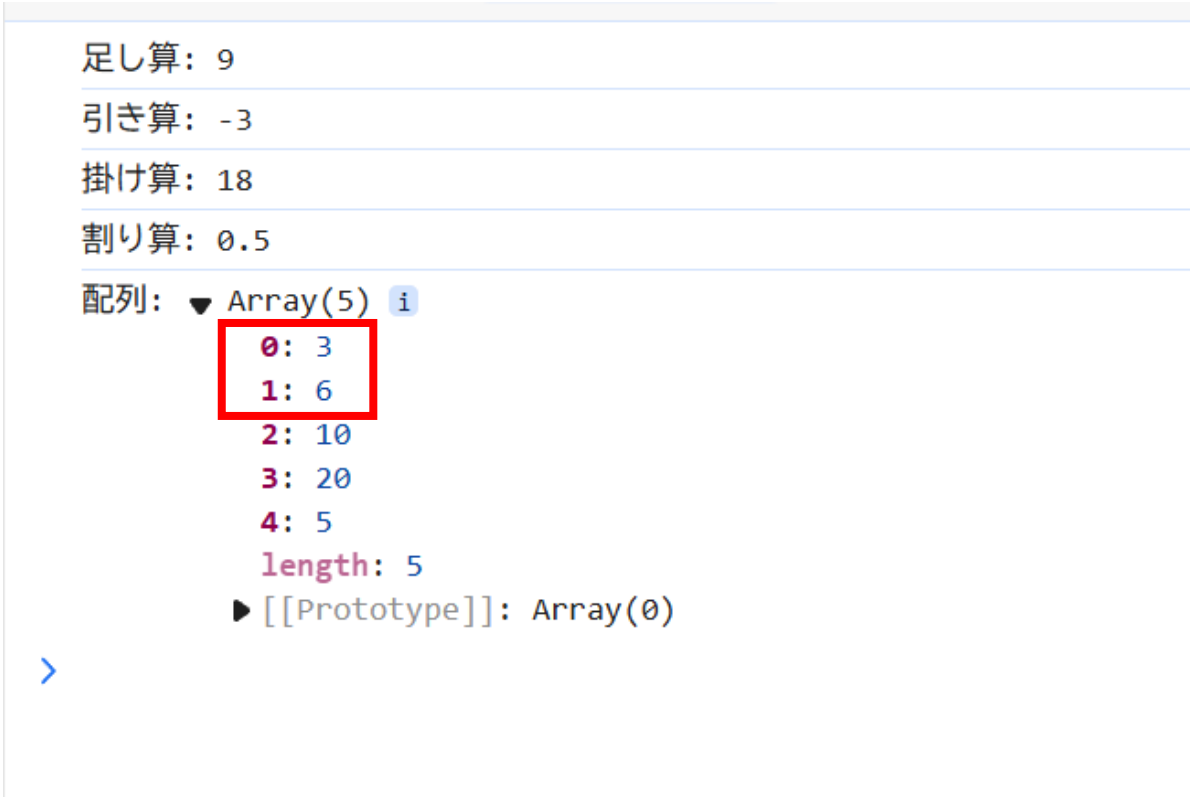
②1次元配列を使ってみよう

コードがかけたら確認してみよう



説明

入力した値は3と6だからこのような表示になっているよ



②1次元配列を使ってみよう

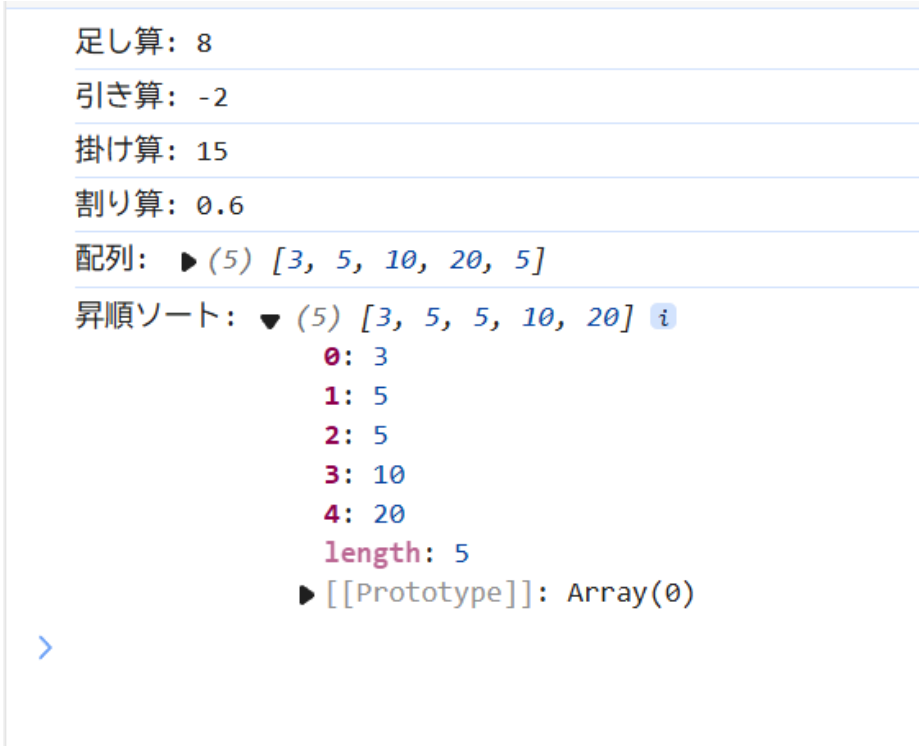
配列の順番を昇順にしよう

※昇順とは数字を小さい順に並べることをいうよ

```
18      const numbers = [number1, number2, 10, 20, 5];
19      console.log("配列:", numbers);
20
21      console.log("昇順ソート:", numbers.slice().sort((a, b) => a - b));
22
```

②1次元配列を使ってみよう

昇順ソートの値が正しい順番になっているか確認しよう



質問

次は降順という大きい数字から表示させるようなコードを書いてみよう

※考えてから次のスライドに進んでみよう！

答え

このようになったらOK

```
20
21      console.log("昇順ソート:", numbers.slice().sort((a, b) => a - b));
22      console.log("降順ソート:", numbers.slice().sort((a, b) => b - a));
23
24  } else {
```

質問

次は5以上の数字を表示させるようなコードを書いてみよう

※考えてから次のスライドに進んでみよう！

答え

このようになったらOK

```
20
21      console.log("昇順ソート:", numbers.slice().sort((a, b) => a - b));
22      console.log("降順ソート:", numbers.slice().sort((a, b) => b - a));
23      console.log("5以上の数値:", numbers.filter(num => num >= 5));
24
```

質問

次は数字の値の合計値を表示させるようなコードを書いてみよう

※考えてから次のスライドに進んでみよう！

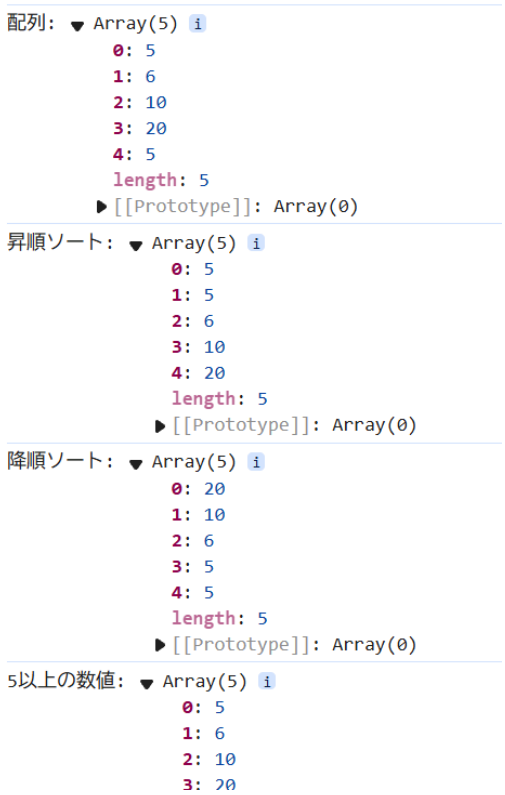
答え

このようになったらOK

```
20
21      console.log("昇順ソート:", numbers.slice().sort((a, b) => a - b));
22      console.log("降順ソート:", numbers.slice().sort((a, b) => b - a));
23      console.log("5以上の数値:", numbers.filter(num => num >= 5));
24      console.log("全要素の合計:", numbers.reduce((sum, num) => sum + num, 0));
25
```

②1次元配列を使ってみよう

出力がこのようになったらOK



目次

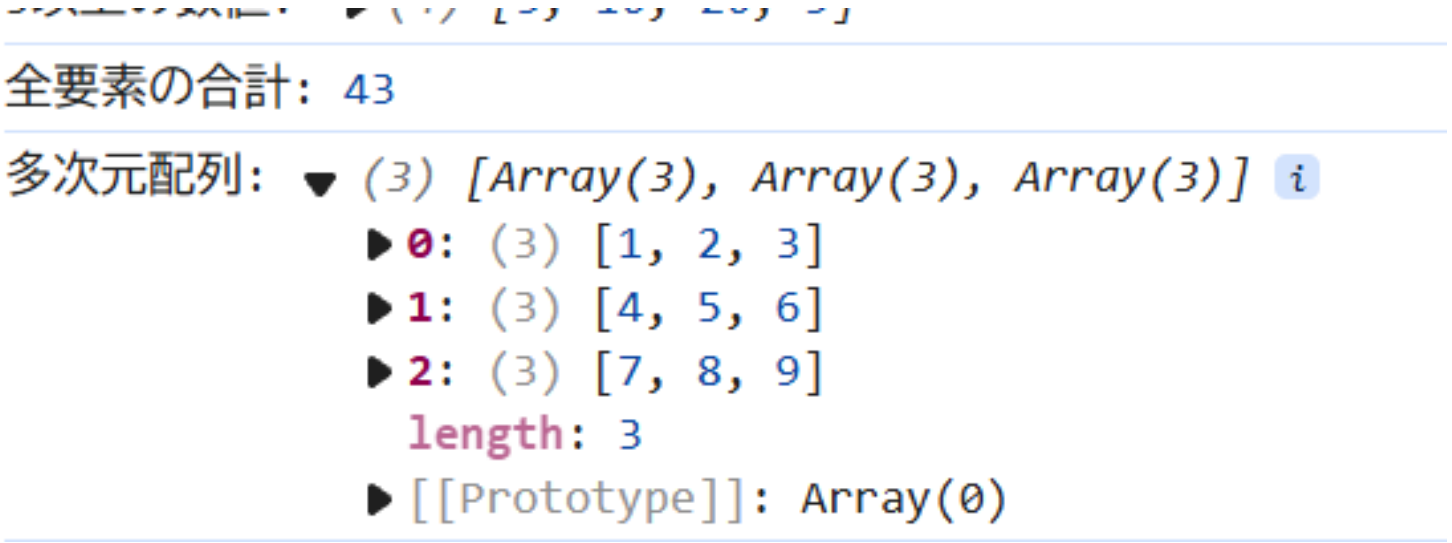
①JSで計算をしてみよう

②1次元配列を使ってみよう

③多次元配列を使ってみよう

③多次元配列を使ってみよう

このページでは配列のちょっとした応用について理解する

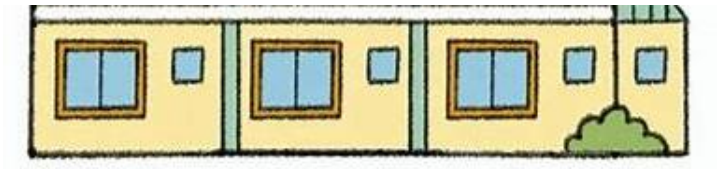


説明

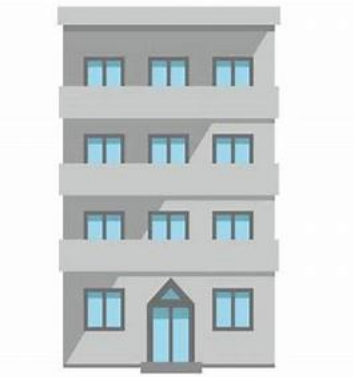
「多次元配列」:配列の中に配列を書いていくこと

配列のイメージ）

1 次元配列
配列[3]



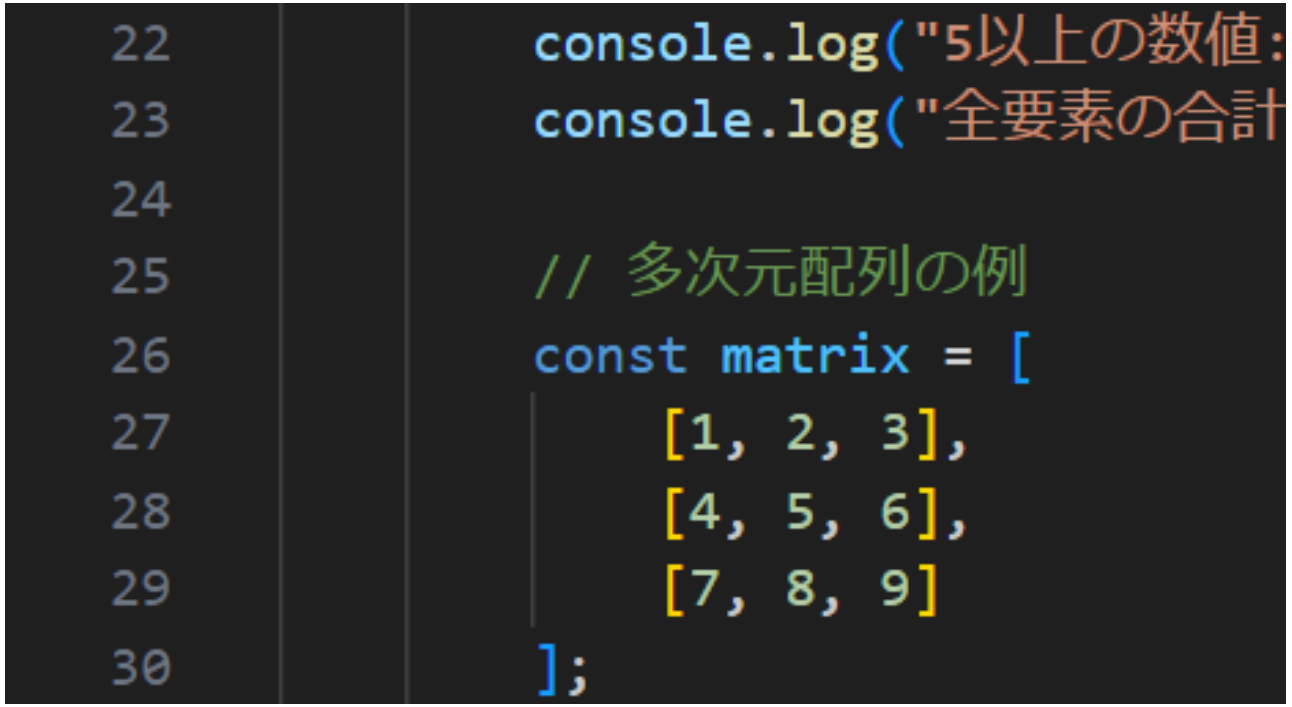
2次元配列
配列[4][3]



3次元配列
配列[3][3][4]

③多次元配列を使ってみよう

多次元配列を作ろう



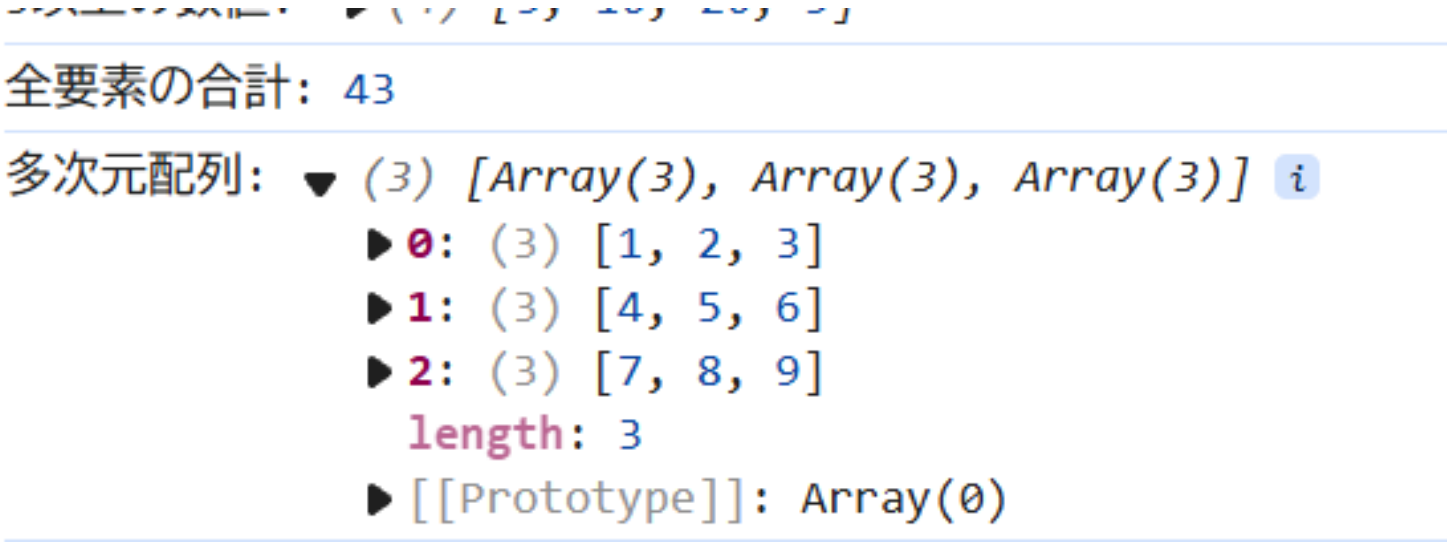
③多次元配列を使ってみよう

出力してみよう

```
26      const matrix = [
27          [1, 2, 3],
28          [4, 5, 6],
29          [7, 8, 9]
30      ];
31      console.log("多次元配列:", matrix);
```

③多次元配列を使ってみよう

このようになったらOK



③多次元配列を使ってみよう

特定の配列を出力できるようにしよう

```
31      console.log("多次元配列:", matrix);
32
33      console.log("特定の要素 (2行目, 3列目):", matrix[1][2]); // 6
34
```

③多次元配列を使ってみよう

次にfor文を使って配列を出力してみよう

```
34
35      // 多次元配列を使ったループ
36      console.log("多次元配列の全要素を表示:");
37      for (let row of matrix) {
38          for (let value of row) {
39              console.log(value);
40          }
41      }
```

お疲れさまでした

テキストは終了です。
あとは自分なりにアレンジを付け加えていこう！