9章 配列を理解しよう

たくさんのデータをまとめて扱うことができる配列について解説します。

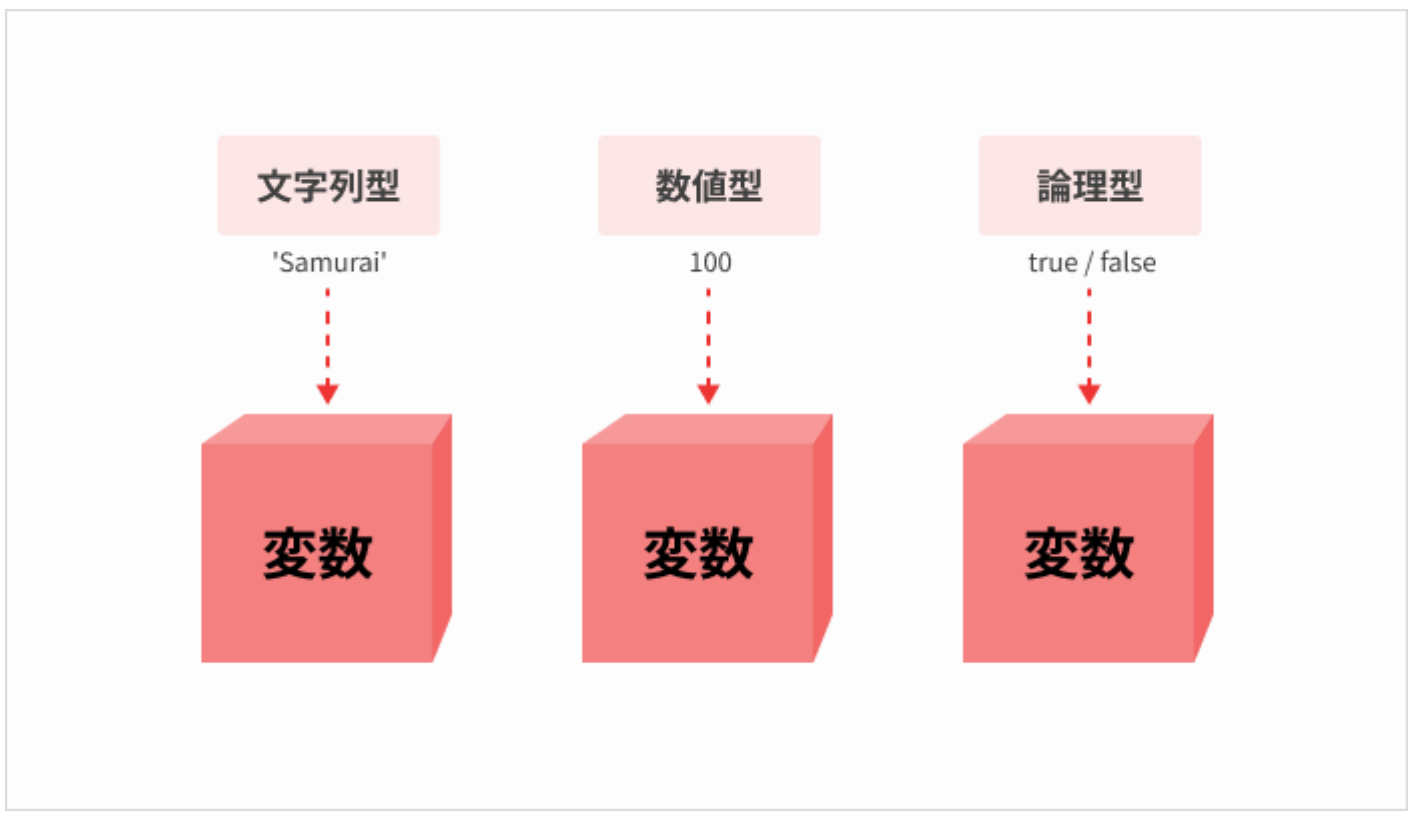
90分 - 読了

9.1 本章の目標

本章では以下を目標にして学習します。

- 配列とは何か、概要をつかむこと
- 配列の作り方、使い方を知ること
- 実際に配列を使ってみること

これまでは、変数や定数に1つのデータしか入れていませんでした。

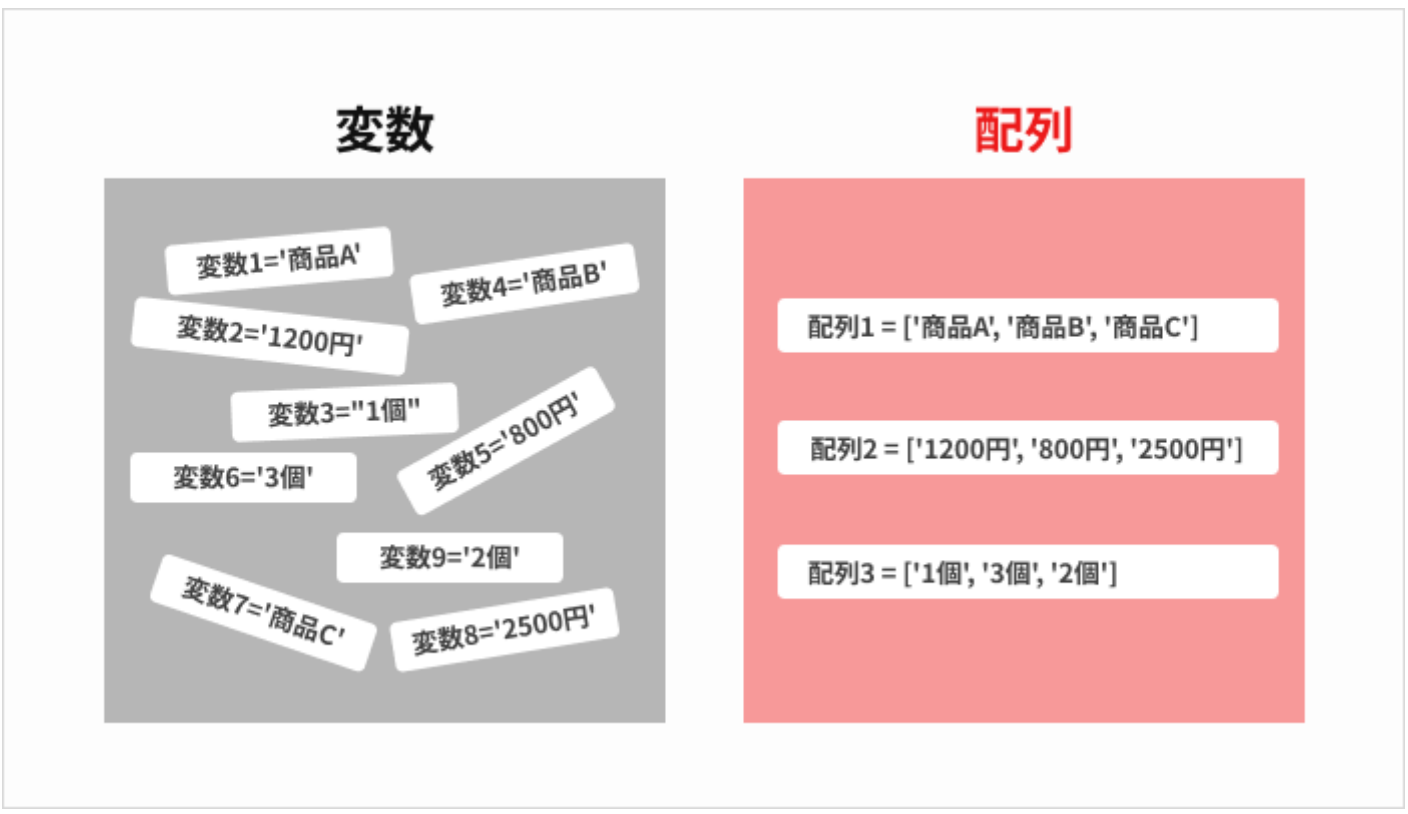


しかし、複数のデータをまとめて管理したいケースもあります。

例えば、Amazonのようなショッピングサイトで買い物をするシーンをイメージしてみてください。欲しい商品を見つけたら、まずはカートに追加するはずです。

そしてカートの中身を見ると、**商品名・価格・数量**が表示されています。

これらのデータは変数や定数で1つずつバラバラに管理するよりも、1つのまとまったデータとして管理したほうが楽です。

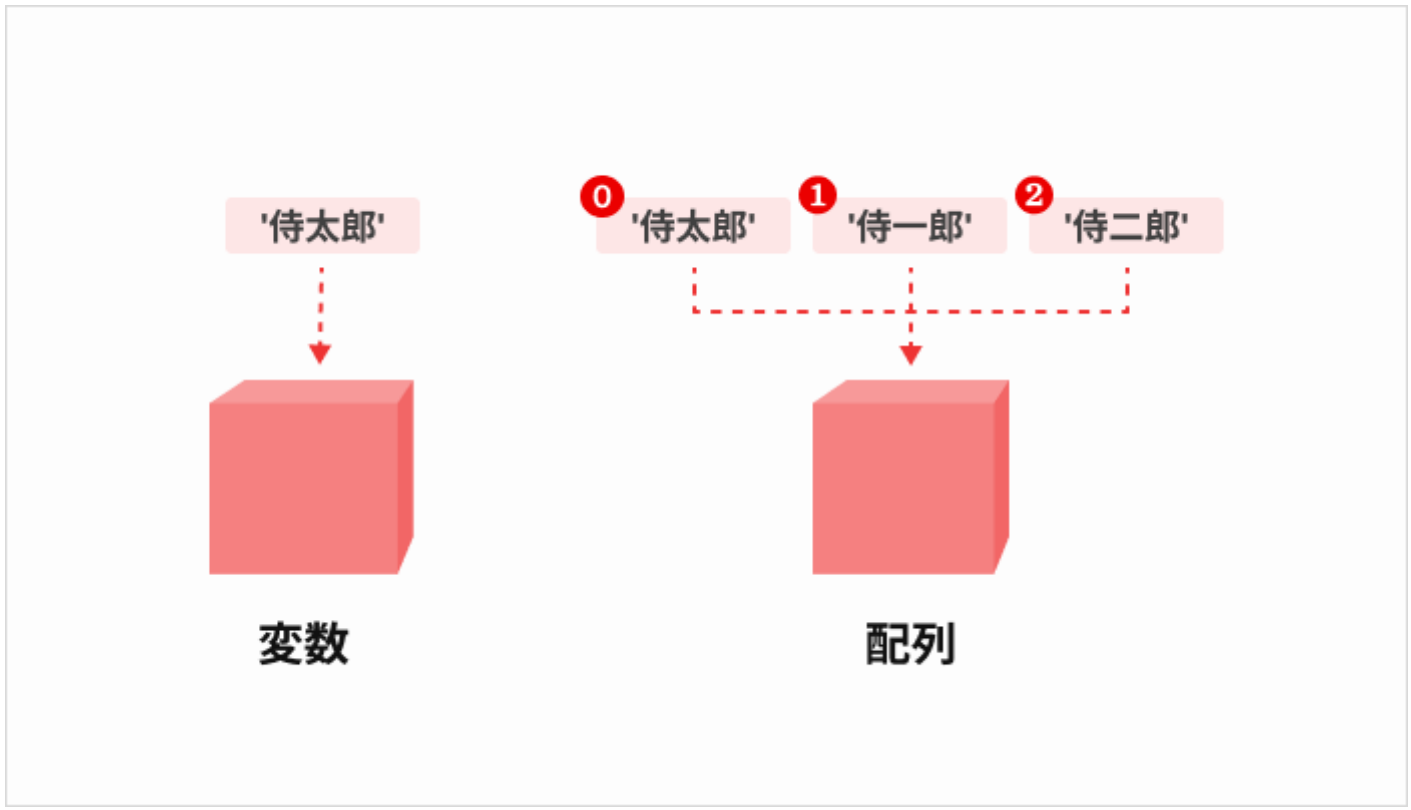


このように、複数のデータを1つにまとめたいときに使われるのが配列です。プログラミングでは配列を当たり前のように使うので、本章でしっかりと使い方を覚えましょう。

9.2 配列とは

配列とは簡単にいえば、**複数のデータのまとまり**のことです。

配列を使うことで、以下のように複数のデータを1つにまとめられます。



9.3 配列の作り方・使い方

配列を作るには、以下のように `[値1，値2，値3，……]` と書きます。なお、配列には似たような値が複数入っているため、一般的に配列名は複数形にします。

JSファイル（見本）

```
const userNames = ['侍太郎', '侍一郎', '侍二郎', '侍三郎', '侍四郎'];
const userAges = [36, 33, 29, 25, 22];

```

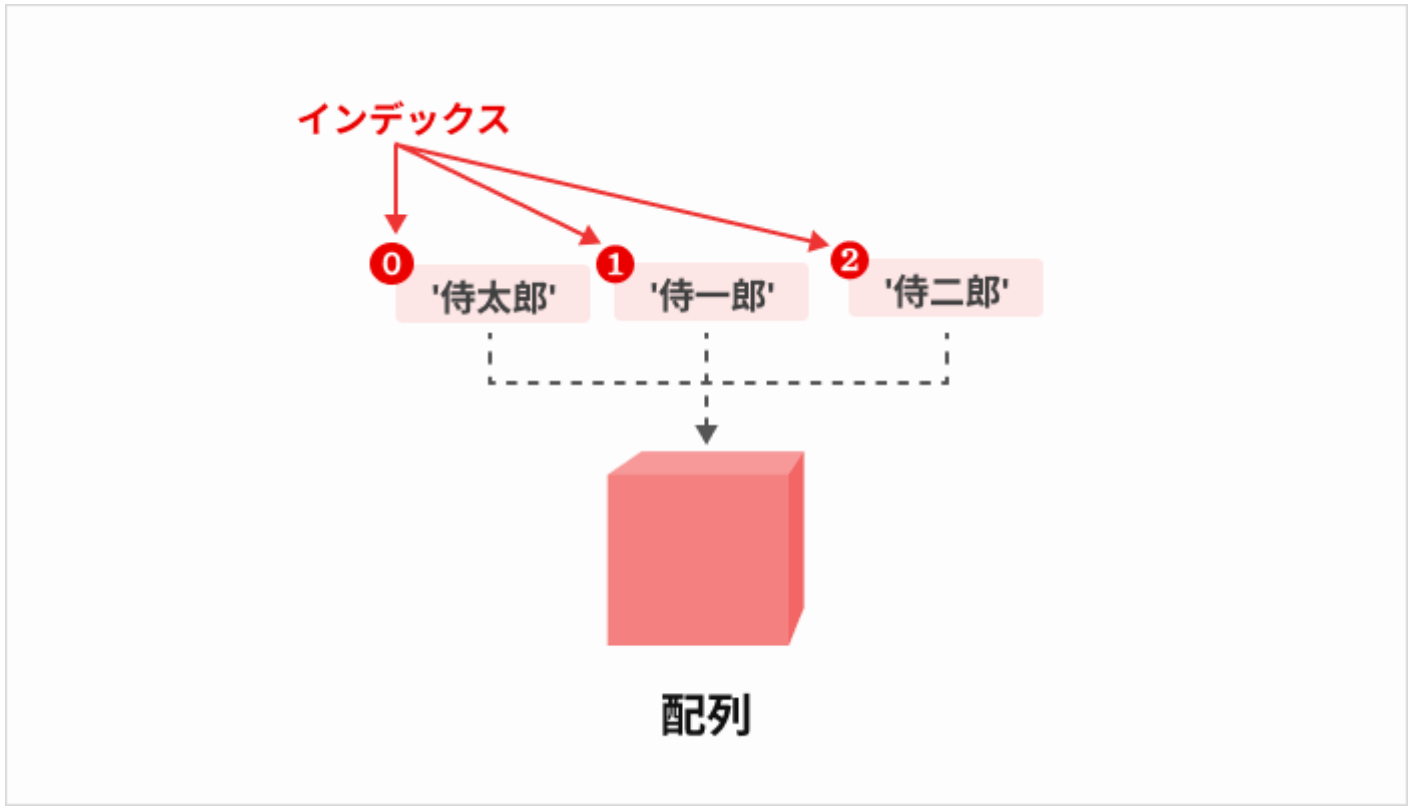
このように、配列は `[]` の中にカンマ区切りで順番に値を入れていきます。

なお、配列に入っているそれぞれの値のことを、**要素**といいます。

要素を取り出す方法

では、例えば「配列 `userNames` の2番目の要素（上記のコードであれば `'侍一郎'` ）だけを取り出して、画面に表示したい」という場合、どうすればよいでしょうか。

もう一度配列の図を見てみましょう。以下のように、配列の各要素には「0」から順番に番号が振られています。この番号のことを、**インデックス**といいます。



このインデックスを使って `配列名[インデックス]` のように記述することで、好きな要素を簡単に取り出すことができます。例えば、配列 `userNames` の2番目の要素を取り出したい場合は、 `userNames[1]` と記述します。

JSファイル（見本）

```
const userNames = ['侍太郎', '侍一郎', '侍二郎', '侍三郎', '侍四郎'];

// 2番目の要素である「侍一郎」という文字列が出力される
console.log(userNames[1]);

```

このとき、インデックスが「0」から始まる点に注意しましょう。インデックスは「0、1、2、……」と続くので、2番目の要素を取り出したいからといって `userNames[2]` と記述すると、3番目の要素が取り出されてしまいます。

要素を更新・追加する方法

配列の要素を**更新**したり、**追加**したりすることもできます。その際も以下のように、インデックスを使います。

JSファイル（見本）

```
const userNames = ['侍太郎', '侍一郎', '侍二郎', '侍三郎', '侍四郎'];

// 2番目の要素を更新する
userNames[1] = '侍花子';

// 6番目に要素を追加する
userNames[5] = '侍五郎';

```

9.4 配列を使ってみよう

では、実際に配列を使ってみましょう。まずはVisual Studio Codeを開き、 `js` フォルダ内に新しく `array.js` というファイルを作成してください（array＝配列）。

続いて、 array.js を以下のように編集しましょう。配列 userNames の宣言および値の代入をおこなったあとに、要素を更新・追加します。

array.js

```
// 配列の宣言と値の代入を行う
const userNames = ['侍太郎', '侍一郎', '侍二郎', '侍三郎', '侍四郎'];

// 配列の値を出力する
console.log(userNames);

// 2番目の要素を更新する
userNames[1] = '侍花子';

// 6番目に要素を追加する
userNames[5] = '侍五郎';

// 配列の値を出力する
console.log(userNames);

```

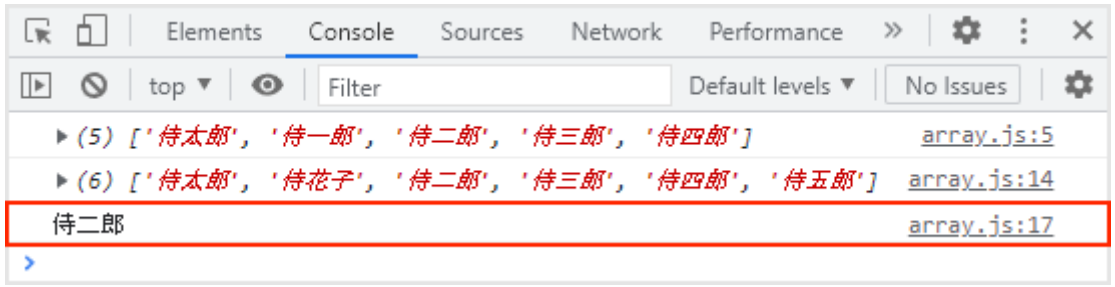
次に index.html を以下のように編集し、読み込むJSファイルを array.js に変更してください。

index.html

```
<!DOCTYPE html>
<html lang="ja">

<head>
  <meta charset="UTF-8">
  <title>JavaScript基礎編</title>
</head>

<body>
-  <script src="js/loop.js"></script>
+  <script src="js/array.js"></script>
</body>

</html>

```

では index.html をブラウザで開き、デベロッパーツールのコンソールを確認してみましょう。以下のように、要素を更新・追加する前後で値が変わっていればOKです。

最後に演習として、配列 userNames の3番目の要素（ '侍ニ郎' ）をコンソールに出力してみてください。以下のようにコンソールに表示されればOKです。



正解のコードは以下のとおりです。

array.js

```
// 配列の宣言と値の代入を行う
const userNames = ['侍太郎', '侍一郎', '侍二郎', '侍三郎', '侍四郎'];

// 配列の値を出力する
console.log(userNames);

// 2番目の要素を更新する
userNames[1] = '侍花子';

// 6番目に要素を追加する
userNames[5] = '侍五郎';

// 配列の値を出力する
console.log(userNames);

+  // 3番目の要素を出力する
+  console.log(userNames[2]);

```

補足：配列の宣言にconstを使う理由

本章では、配列を宣言するときに `const` を使いました。

JSファイル（見本）

```
// 配列の宣言と値の代入を行う
const userNames = ['侍太郎', '侍一郎', '侍二郎', '侍三郎', '侍四郎'];

```

5章の章末でもお伝えしたとおり、「予期せぬ動作を防げる」というメリットがとても大きいので、JavaScriptでは基本的に定数（ `const` ）を使うのが一般的です。

しかし、配列では要素を更新したり追加したりするのに、なぜ `let` ではなく `const` を使うのか、疑問に思った方もいるかもしれません。

ここで、 `let` と `const` の違いを復習しておきましょう。

- `let` ：変数を宣言するときに使う。あとから中身を入れ替えられる（再代入できる）
- `const` ：定数を宣言するときに使う。あとから中身を入れ替えられない（再代入できない）

`const` で定数を宣言したとしても、以下のように配列の要素を更新したり、追加したりすることはできます。なぜなら、要素の更新や追加は再代入にあたらないためです。

JSファイル（見本）

```
// 配列の宣言と値の代入を行う
const userNames = ['侍太郎', '侍一郎', '侍二郎', '侍三郎', '侍四郎'];

// 2番目の要素を更新する
userNames[1] = '侍花子';

// 6番目に要素を追加する
userNames[5] = '侍五郎';

```

ただし以下のように、値を再代入する（丸ごと入れ替える）とエラーが発生します。

JSファイル（見本）

```
// 配列の宣言と値の代入を行う
const userNames = ['侍太郎', '侍一郎', '侍二郎', '侍三郎', '侍四郎'];

// 値を再代入する（丸ごと入れ替える）とエラーが発生する
userNames = ['侍花子', '侍一子'];

```

このように一度宣言した配列に値を再代入する（丸ごと入れ替える）ケースはほとんどないため、配列には `const` を使うのが一般的です。「配列を宣言するときは基本的に `const` を使う」と覚えておきましょう。

本章の学習は以上です。お疲れさまでした。

まとめ

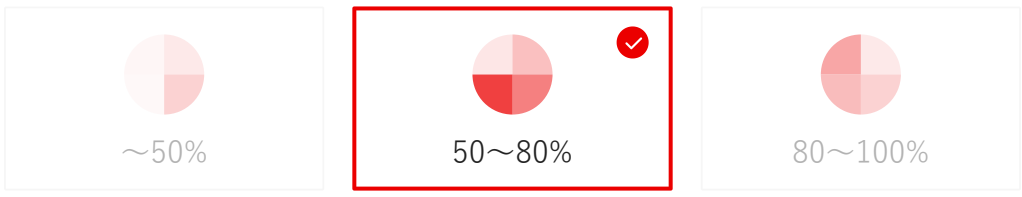
本章では以下の内容を学習しました。

- 配列とは**複数のデータのまとまり**のことである
- 配列は `[値1，値2，値3，……]` のように、 `[]` の中にカンマ区切りで順番に値を入れていく
- 配列に入っているそれぞれの値のことを、**要素**という
- 配列の各要素には、「0」から順番に番号（**インデックス**）が振られている
- `配列名[インデックス]` のように記述すれば、特定の要素を取得・更新・追加できる
- 配列を宣言するときは `const` を使う

次章では、オブジェクトについて学びます。

理解度を選択して次に進みましょう

ボタンを押していただくと次の章に進むことができます



最後に確認テストを行いましょう

下のボタンを押すとテストが始まります。

教材をみなおす　テストをはじめる

前に戻る　11 / 26 ページ　次に進む

< 一覧に戻る　改善点のご指摘、誤字脱字、その他ご要望はこちらからご連絡ください。