4章 変数を理解しよう

データをより便利に扱える変数について解説します。

90分 - 読了

4.1 本章の目標

本章では以下を目標にして学習します。

- 変数とは何か、概要をつかむこと
- 変数の宣言と値の代入について理解すること
- 変数を実際に使ってみること
- 変数名のつけ方ルール（命名規則）を知ること

前章では、JavaScriptに文字列型や数値型といったデータ型が存在することを学びました。では、例えばWebアプリにおいて、ユーザーごとに「おかえりなさい、○○さん」という文字列を表示したい場合を考えてみましょう。

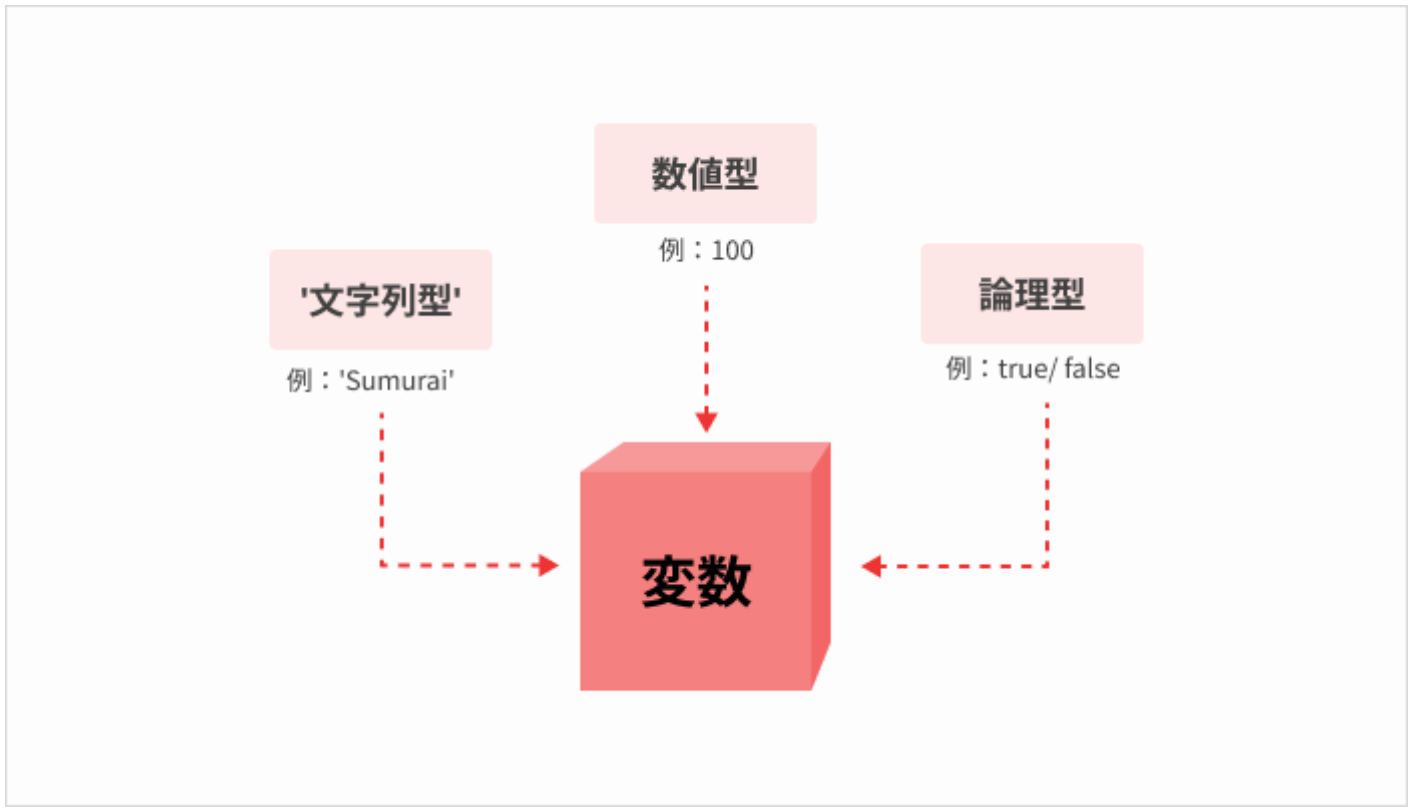
文字列をそのまま記述する場合、「おかえりなさい、侍太郎さん」「おかえりなさい、侍花子さん」など、ユーザーごとにコードを変えなければならず、とても大変です。

そこで大活躍するのが**変数**です。JavaScriptをはじめとするプログラミング言語では、変数を当たり前のように使います。

本章で変数について理解を深め、プログラミング学習の第一歩を踏み出しましょう。

4.2 変数とは

変数とは簡単にいえば、文字列や数値などのデータを入れる箱のようなものです。変数の中身はいつでも入れ替えることができます。



4.3 変数の宣言・値の代入

変数を使うには、変数の**宣言**および値の**代入**を行う必要があります。宣言と代入の意味はそれぞれ以下のとおりです。

- 宣言：「こんな名前の変数を使いますよ」と宣言すること
- 代入：宣言した変数に実際の値（データ）を入れること

変数の宣言と値の代入は同時に行うこともできますし、別々に行うこともできます。

変数を使ってみよう

では、実際に変数を使ってみましょう。

JSファイルの作成

まずはVisual Studio Codeを開き、 `js` フォルダ内に新しく `variable.js` というファイルを作成してください（variable＝変数 ※覚える必要はありません）。

変数の宣言

続いて `variable.js` を以下のように編集し、 `userName` という名前の変数を宣言しましょう。変数を宣言するには `let` を記述します（なぜ変数名が `username` ではなく `userName` なのか、理由はこのあと5節で解説します）。

variable.js

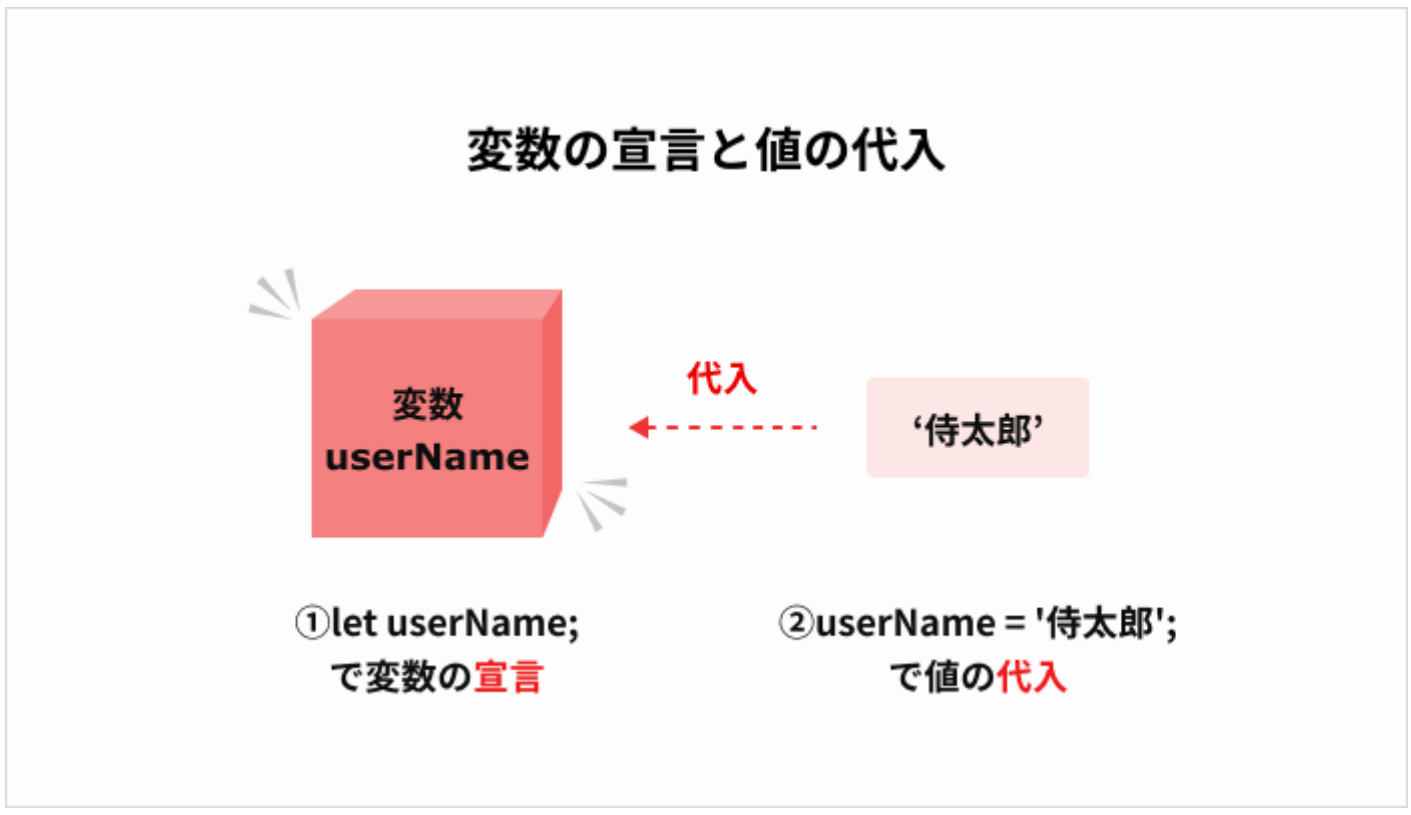
```
+ // 変数の宣言
+ let userName;
```

値の代入

次に `variable.js` を以下のように編集し、変数 `userName` に値を代入しましょう。値を代入するときは、 `let` を記述する必要はありません。 `let` は宣言するときにのみ必要です。

variable.js

```
// 変数の宣言
let userName;

// 値の代入
userName = '侍太郎';
```



変数の宣言・値の代入

続いて、変数の宣言と値の代入を同時にやってみましょう。 `variable.js` を以下のように編集してください。

variable.js

```
// 変数の宣言
let userName;

// 値の代入
userName = '侍太郎';

// 変数の宣言・値の代入
let userNumber = 55;
```

コンソールへの出力

では、変数の中身をコンソールに出力してみましょう。 `variable.js` を以下のように編集してください。

variable.js

```
// 変数の宣言
let userName;

// 値の代入
userName = '侍太郎';

// 変数の宣言・値の代入
let userNumber = 55;

// コンソールへの出力
console.log(userName);
console.log(userNumber);

```

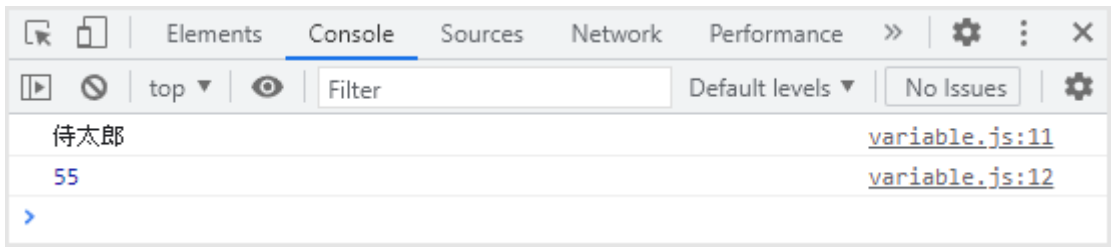
続いて、前章で作成した `index.html` を以下のように編集し、読み込むJSファイルを `variable.js` に変更してください。

index.html

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

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

<body>
-   <script src="js/data.js"></script>
+   <script src="js/variable.js"></script>
</body>

</html>

```

実行結果

`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、変数の中身が表示されていればOKです。



4.4 変数の中身を入れ替えてみよう

前述のとおり、変数の中身はあとから入れ替えることができます。実際にやってみましょう。

値の再代入

`variable.js` を以下のように編集してください。変数に値を代入し直す（再代入する）だけで、変数の中身を入れ替えることができます。最初に代入したときと同様、 `let` を記述する必要はありません。

variable.js

```
// 変数の宣言
let userName;

// 値の代入
userName = '侍太郎';

// 変数の宣言・値の代入
let userNumber = 55;

// コンソールへの出力
console.log(userName);
console.log(userNumber);

+ // 値の再代入
+ userName = '侍花子';
+ userNumber = 88;

```

コンソールへの出力

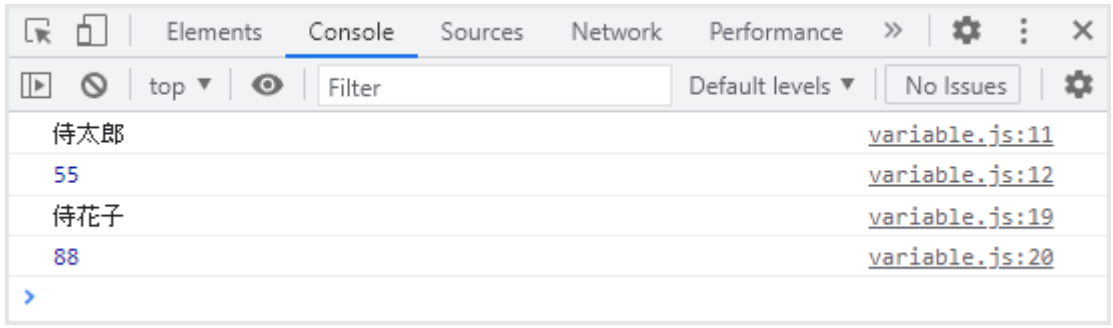
では、値を再代入した変数の中身をコンソールに出力してみましょう。 `variable.js` を以下のように編集してください。

variable.js

```
// 変数の宣言
let userName;

// 値の代入
userName = '侍太郎';

// 変数の宣言・値の代入
let userNumber = 55;

// コンソールへの出力
console.log(userName);
console.log(userNumber);

// 値の再代入
userName = '侍花子';
userNumber = 88;

+ // コンソールへの出力
+ console.log(userName);
+ console.log(userNumber);

```

実行結果

`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、変数の中身が入れ替わっていればOKです。



同じ変数（ `userName` と `userNumber` ）を出力しているにもかかわらず、値を再代入したあとは中身が変わり、出力される内容も変わることがおわかりいただけたかと思います。

4.5 変数名のつけ方ルール（命名規則）

変数の使い方がわかったところで、変数名のつけ方を学びましょう。プログラミング言語では、名前のつけ方のルールのことを**命名規則**といいます。

変数は基本的にどのような名前をつけても動作します。しかし、意味のわからない適当な名前をつけてしまうと、あとから見たときに「この変数は何のためにあるんだっけ」となりがちです。

また、チームで開発するうえでも「誰が見ても内容がわかる変数名にする」というのが基本です。

そこで、一般的な変数名の命名規則を3つ覚えておきましょう。

1. 変数名はキャメルケースで記述する
2. 変数の中身がわかるような名前にする
3. 予約語は使えない

キャメルケースと予約語については後述します。順番に見ていきましょう。

1. 変数名はキャメルケースで記述する

先ほど変数を宣言したときに、 `userName` や `userNumber` という名前をつけました。「なぜ `username` や `user-name` じゃないんだろう」と思った方もいるかもしれません。これは**キャメルケース**という記法にもとづいています。

キャメルケースとは、アルファベットで複合語（複数の単語から成り立つ語）を記述する際に、各単語の先頭を大文字にする記法のことです。

JavaScriptではキャメルケースで変数名をつけることが一般的なので、この書き方に慣れていきましょう。

補足1：キャメルの由来

キャメルは英語でcamelと書きますが、これは「ラクダ」という意味です。大文字の部分がラクダのこぶに見えることから、この名前がつけられました。

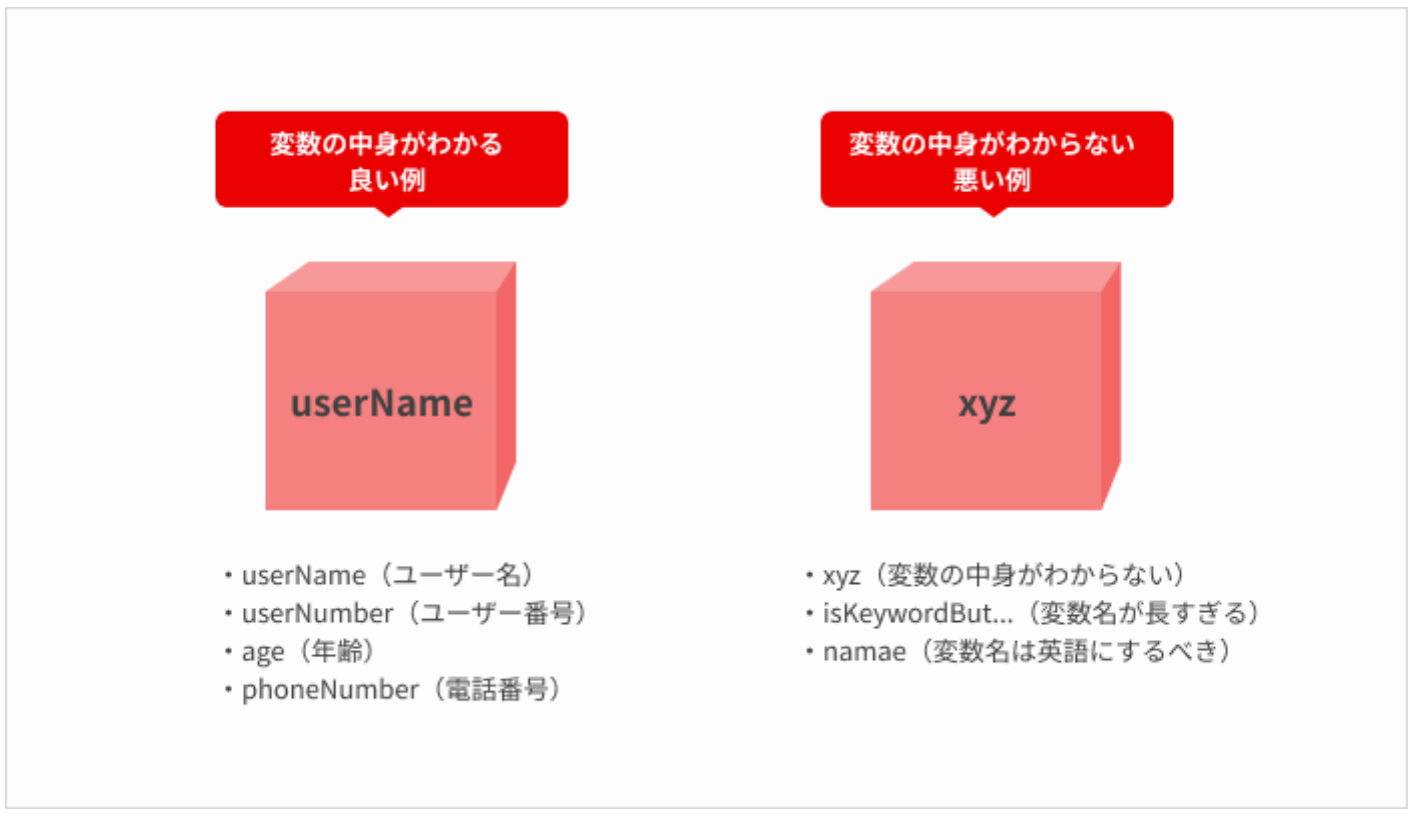
補足2：ローワーキャメルケースとアッパーキャメルケース

1つ目の単語だけ小文字にするローワーキャメルケースと、1つ目の単語も大文字にするアッパーキャメルケース（パスカルケースともいう）があるのですが、単にキャメルケースという場合はローワーキャメルケースを指すことが多いです。

- ローワーキャメルケース： `userName` 、 `userNumber` 、 `camelCase`
- アッパーキャメルケース（パスカルケース）： `UserName` 、 `UserNumber` 、 `PascalCase`

JavaScriptではローワーキャメルケースが一般的です。

2. 変数の中身がわかるような名前にする



変数を宣言するときは、その中身がわかるような名前をつけましょう。以下はその一例です。

- `userName` （ユーザー名）
- `userNumber` （ユーザー番号）
- `age` （年齢）
- `phoneNumber` （電話番号）
- `alertMessage` （警告メッセージ）

逆に、悪い例も掲載しておきます。

- `xyz` （変数の中身がわからない）
- `isKeywordButNothingOrSomething` （変数名が長すぎる）

- `namae` （変数名は英語にするべき）

3. 予約語は使えない

本節の冒頭で「変数は基本的にどのような名前をつけても動作します」と述べましたが、例外があります。それが**予約語**です。

予約語とは、プログラミング言語において、あらかじめ意味のある文字列として定義されている単語のことです。噛み砕いていえば、「それはもう役割が決まっているから好きに使わないでね」という単語です。

例えば、先ほど変数を宣言するときに記述した `let` も予約語です。よって、変数に `let` という名前をつけることはできません。

その他の予約語についてはMDNのリファレンスを参照してください（覚える必要はないので参考程度でOKです）。

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。

- 変数とは簡単にいえば、文字列や数値などのデータを入れる箱のようなものである
- 変数の宣言・値の代入
  - 宣言：「こんな名前の変数を使いますよ」と宣言すること（例： `let userName;` ）
  - 代入：宣言した変数に実際の値を入れること（例： `userName = '侍太郎';` ）
- 変数に値を再代入すれば、変数の中身を入れ替えることができる
- 変数名の付け方ルール（命名規則）
  - 1. 変数名はキャメルケースで記述する
  - 2. 変数の中身がわかるような名前にする
  - 3. 予約語は使えない

次章では、定数について学びます。

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　4 / 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。