5章 定数を理解しよう

定数の使いどころや、変数との違いについて解説します。

90分 - 読了

5.1 本章の目標

本章では以下を目標にして学習します。

- 定数とは何か、概要をつかむこと
- 定数の宣言と値の代入について理解すること
- 定数を実際に使ってみること

前章では、「変数とは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り、中身はいつでも入れ替えられる」ということを学びました。

プログラミングをしていると、この一見便利な「中身をいつでも入れ替えられる」という性質が、予期せぬ動作につながってしまうことがあります。

例えば「変数○○の中身には△△が入っていると思っていたが、いつの間にか□□に入れ替わっていた」というケースです。

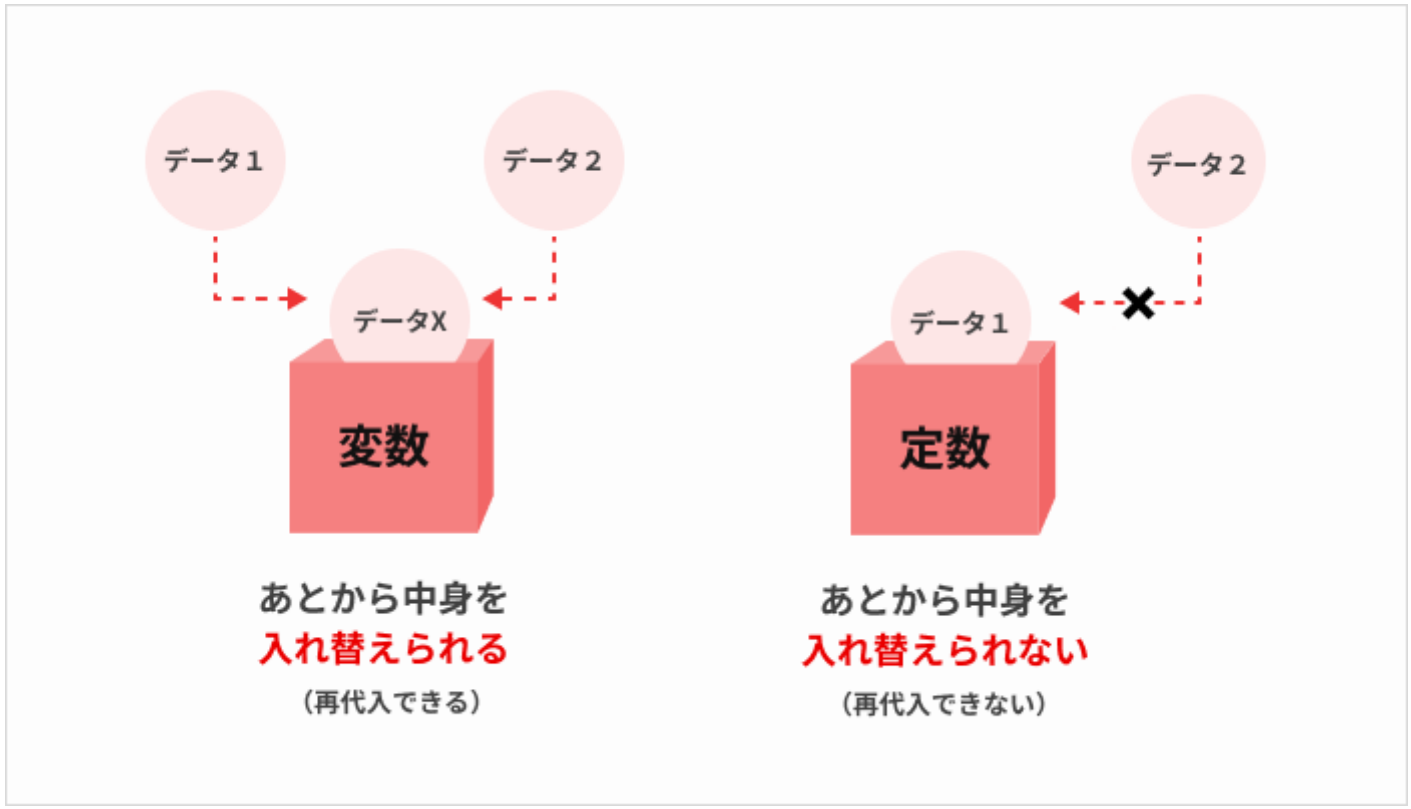
こういったケースを防ぎたいときに便利なのが、**定数**です。変数と定数を使い分けることで、よりミスの少ないコードを書けるようになります。

本章で定数について理解を深め、変数と定数を使い分けられるようになりましょう。

5.2 定数とは

定数とは簡単にいえば、「あとから中身を入れ替えられない変数」のことです。

- 変数：あとから中身を入れ替えられる（再代入できる）
- 定数：あとから中身を入れ替えられない（再代入できない）



こうして見ると、中身を入れ替えられない定数は不便だと思うかもしれません。しかし、「**予期せぬ再代入を防げる**」という大きなメリットがあります。

例えばショッピングサイトの開発において、一律で適用される送料があったとします。この送料を変数で宣言してしまうと、一度その変数名を使ったことを忘れ、また別の値を再代入してしまうおそれがあります。

意図せず送料が変わってしまったら、大問題です。しかし、送料を定数で宣言しておけば、この問題をあらかじめ防ぐことができます。

他にも例えば消費税の税率など、あらかじめ定数で宣言しておけば、増税したときに定数の値を修正するだけでプログラム全体の税率を一括で変更できます。

このように、「一律で値が決まっている**固定値**には定数を使う」と覚えておきましょう。

5.3 定数の宣言・値の代入

定数を使うには、定数の宣言および値の代入を行う必要があります。この点は変数と同様です。宣言と代入の意味を復習しておきましょう。

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

定数は一度代入したら終わりなので、宣言と代入を同時に行うのが一般的です。

定数を使ってみよう

では、実際に定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 `constant.js` というファイルを作成してください。

定数の宣言・値の代入

続いて `constant.js` を以下のように編集し、定数の宣言と値の代入を同時に行いましょう。前章で変数を宣言したときは `let` を記述しましたが、定数を宣言するときには `const` を記述します。なお、constは定数を意味するconstantの略です。

constant.js

```
// 定数の宣言・値の代入
const shippingFee = 500;

```

- shipping fee＝送料（※覚える必要はありません）

コンソールへの出力

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

constant.js

```
// 定数の宣言・値の代入
const shippingFee = 500;

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

```

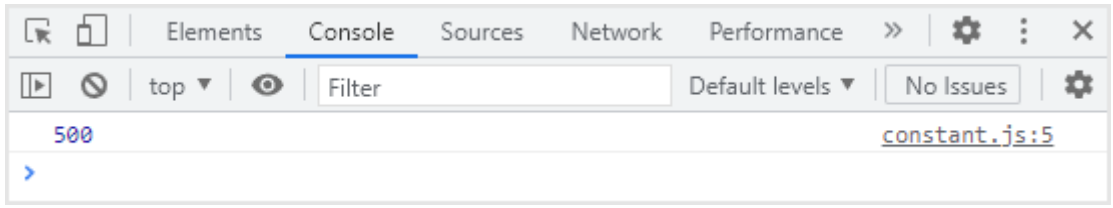
続いて `index.html` を以下のように編集し、読み込むJSファイルを `constant.js` に変更してください。

index.html

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

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

<body>
-   <script src="js/variable.js"></script>
+   <script src="js/constant.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、定数の中身が表示されていればOKです。



5.4 試しに定数の中身を入れ替えてみよう

前述のとおり、定数はあとから中身を入れ替えられません。それを確かめるために、試しに定数の中身を入れ替えてみましょう。

値の再代入（エラー）

`constant.js` を以下のように編集し、値を再代入してください。

constant.js

```
// 定数の宣言・値の代入
const shippingFee = 500;

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

+ // 値の再代入（エラー）
+ shippingFee = 800;

```

コンソールへの出力

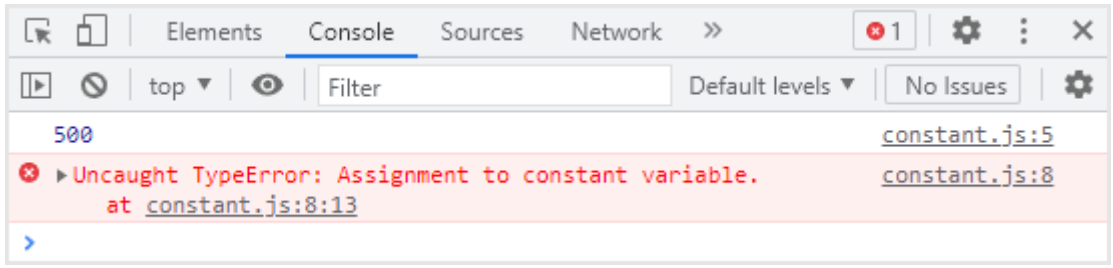
では、定数の中身をコンソールに出力してみましょう。 `constant.js` を以下のように編集してください。

constant.js

```
// 定数の宣言・値の代入
const shippingFee = 500;

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

// 値の再代入（エラー）
shippingFee = 800;

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

```

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、定数 `shippingFee` の値は変わらず `500` のままです。また、定数に値を再代入しようとしたため、エラーが発生しています。



なお、「Uncaught TypeError: Assignment to constant variable.」を翻訳すると、「未補足の型エラー：定数への代入」という意味です。定数に値を再代入しようとすると、このエラー文が表示されます。

前章で学んだ変数の場合は再代入が可能でした。しかし、定数の場合は値を再代入しようとするとすぐにエラーで知らせてくれるので、予期せぬ再代入を防ぐことができます。

5.5 変数と定数の使い分け

前章から変数と定数について学んできましたが、「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、「基本的には定数（ `const` ）を使う」のが一般的です。本教材でも、今後は基本的に `const` で統一します。

ただし、再代入が必要なケース（8章で学ぶfor文のカウンタ変数など）でのみ、変数（ `let` ）を使います。

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。

- 定数とは簡単にいえば、「あとから中身を入れ替えられない変数」のことである
- 定数には「**予期せぬ再代入を防げる**」というメリットがある
- あとから中身を入れ替えたくない**固定値**には、定数を使う
- 定数に値を再代入しようとすると、エラーが発生する
- 再代入が必要なケースを除き、基本的には定数（ `const` ）を使うのが一般的である

次章では、条件分岐のif文について学びます。

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

～50%　50～80%　80～100%

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　5 / 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。