JavaScript講座 第２回

2026/01/22

Tam

第2回／1月22日(木)

「JavaScript文法の基礎」

- 変数、定数、配列
- 論理演算と条件分岐
- 繰り返し命令
- エラー対策

変数宣言

`let` で変数を宣言します。

※「宣言」とは、「この名前の変数を使用する」とコンピューターに伝えることです。

```
let greeting;
```

※命令の終わりには、記号「;」を入れるようにしてください。

代入

「変数 = 値;」と書くことで、右辺の値を左辺の変数に代入できます。

```
greeting = "Hello";
```

宣言とともに代入

以下のようにすると、宣言と同時に変数に値を代入して初期化できます。

```
let greeting = "Hello";
```

再宣言

すでに宣言されている変数は、再度宣言することが出来ません。

```
// ここはコメントアウトです。前回と同様、入力しないでください。
let greeting = "Hello"; // 変数宣言
greeting = "Good Evening"; // 代入はできる
let greeting; // ここでエラー
```

実践

ここまでの内容を実践してみましょう。

```
<html>
  <body>
    <p id="greeting">Hello, World.</p>
  </body>
  <script type="text/javascript">
    // 変数 greeting を宣言し、 "Hello" を代入
    let greeting = "Hello";
    // 変数 greeting の内容を表示
    console.log(greeting);
    // 変数 greeting に "Good Evening" を代入
    greeting = "Good Evening";
    // 再宣言を行うため、以下でエラーが発生
    let greeting;
  </script>
</html>
```

定数

定数の概念もあります。定数は `const` で宣言します。

```
const tax = 0.10;
```

定数は、 ***再代入のできない変数*** として、変数と同様に扱えます。

計算

JavaScript では、四則演算が使えます。

足し算（+）、引き算（-）は記号どおりですが、
掛け算は「*（アスタリスク）」、割り算は「/（スラッシュ）」を使います。

また、除算（割った余り）として「%」記号も使えます。（よく使います。）

```
const tax = 0.10; // 消費税率
let price = 100; // 価格
let taxfee = price * tax; // 消費税を計算して、変数 taxfee に代入
let total_price = price + taxfee; // 税込価格を計算して、変数 total_price に代入
console.log(total_price); // 計算した値を表示
```

繰り返し命令（while文）

```
while (式) {
    文;
}
```

と書くことにより、「文」のプログラムを繰り返し実行させることが出来ます。

皆さんにはカレンダーを作ってもらう目標がありますので、
まずは１月１日から１月３１日までを表示する例を示します。

```
let i = 1; // 変数 i を宣言
while (i <= 31) { // ここから繰り返し
    // １月？日を表示
    console.log("1月" + i + "日");
    i = i + 1; // 変数 i を１つ増やす
}
```

比較演算子

先程の `i <= 31` の部分では ***i が31以下だったら*** という条件を見ています。

このような条件を見るプログラムを ***比較演算子*** と言い、以下のような種類があります。

- `i < 31` i が 31未満、 `i <= 31` i が 31以下
- `i > 31` i が 31より大きい、 `i >= 31` i が 31以上
- `i === 31` i が 31と等しい、 `i !== 31` i が 31でない

※`i == 31` 、 `i != 31` などもありますが、この講座では出来るだけ使わないでください。

インクリメント、デクリメント

先程の例で出てきた `i = i + 1;` という文は、プログラムならではの書き方です。

大変よく使うので、 `i += 1;` などと省略でき、もっと省略して `i++;` などと書くことも出来ます。

※「インクリメント」と呼びます。

`i = i - 1;` もよく使われ、こちらも `i -= 1` と省略でき、さらに省略して `i--;` として書けます。

※「デクリメント」と呼びます。

繰り返し命令（for文）

先程の例は、 for文を使用すると、もっとコンパクトに書けます。

```
for (let i = 1; i <= 31; i += 1) { // ここから繰り返し
    // １月？日を表示
    console.log("1月" + i + "日");
}
```

実際には、 for文を用いた繰り返しのほうがたくさん使われるので、
どちらでも書けるようにしておきましょう。

曜日を付加したい

除算を用いて、数字で曜日を出してみましょう。

計算機は、数字の扱いが得意なので、０を日曜日、６を土曜日を表すものとして表示させてみましょう。

今月の１月１日は木曜日でしたので、以下のように出来ます。

```
let wday = 4; // 木曜日
for (let i = 1; i <= 31; i++) { // ここから繰り返し
    // １月？日を表示
    console.log("1月" + i + "日(" + wday + ")");
    wday++; // 曜日をインクリメント
    wday = wday % 7; // 7 で割った余りを計算
}
```

配列

たくさんの変数をまとめて扱うことができます。

```
let fruits = ["orange", "apple", "strawberry"];
console.log(fruits[0]); // 0番目の orange を表示
console.log(fruits[2]); // 2番目の strawberry を表示
furits[2] = "pine"; // 2番目に pine  を代入
console.log(furits[2]); // 2番目を表示
```

※JavaScript では、配列の先頭は 0 番目から始まります。

※配列の先頭は、言語によって、 0番目だったり、 1番目だったりします。

曜日の表示

配列を利用して、曜日を数字ではなく、`日`〜`土`で表示してみましょう。

```
// 曜日の文字列を定数で定義
const wdays = ["日", "月", "火", "水", "木", "金", "土"];
let wday = 4; // 木曜日
for (let i = 1; i <= 31; i++) { // ここから繰り返し
    // 1月？日を表示
    console.log("1月" + i + "日(" + wdays[wday] + ")");
    wday++; // 曜日をインクリメント
    wday %= 7; // 7 で割った余りを計算
}
```

文字列の足し算

文字も足し算することが出来ます。

```
let greeting;
greeting = "Hello, " + "everyone";
console.log(greeting);
```

行で表示

文字列の足し算を利用して、カレンダーを１行で表示してみましょう。

```
let line = "";
for (let i = 1; i <= 31; i++) { // ここから繰り返し
    line += " "; // 数字の前に、区切りのためにスペースを挿入
    line += i; // 日付を追加
}
console.log(line);
```

条件文技

ある条件のときにだけプログラムを実行させることが出来ます。

```
const i = 16777216;
if (i % 2 === 0) {
    // i を 2 で割った余りが 0 と等しいとき
    console.log("偶数");
} else {
    // それ以外のとき
    console.log("奇数");
}
```

※「等しい」を表す式として、「==」と「===」がありますが、この講座では「===」を使用してください。

実践

日曜日で折り返して、カレンダーっぽくしてみましょう。

```
let wday = 4; // 木曜日
let line = "";
for (let i = 1; i <= 31; i++) { // ここから繰り返し
    if (wday === 0) {
        // 日曜なので、行を折り返す。
        console.log(line); // line を出力
        line = ""; // line の内容を初期化
    }
    line += " "; // 数字の前に、区切りのためにスペースを挿入
    line += i; // 日付を追加
}
// 最後に line の残った内容を出力
console.log(line);
```