JavaScriptで学ぶ プログラミング入門 丸一日コース 問題集

第４版

SAMPLE

武舎広幸 著

はじめに

この問題集は、YouTubeで公開している動画『YouTube版 JavaScriptで学ぶ プログラミング入門 丸１日コース』をご覧になった方（現在ご覧になっている方）のための問題集として作成しました。この講座のサポートページ（https://musha.com/sc/jsut）には、上記の動画や参考資料などが公開されていますのでご覧ください。

詳細はサポートページや動画に譲りますが、丸一日（約5時間）を使って、プログラミングとは何かを体感していただき、JavaScriptの基本を学びながら、プログラミング技術の習得に必要な「感覚」を身につけていただくために筆者がゼロから開発した講座です。小手先の技術だけではなく、プログラミングの基本がしっかり身に付くように、重要なポイントをもれなく解説しており、2014年10月以来、ストアカ、connpassなどのサイトを通して2,000人以上の方に受講いただいています。

この問題集は上記の講座の「実習」で使われてきた問題を1冊にまとめたものです。非常に単純な問題から、ある程度高度な問題まで、80問以上の練習問題を解くことで、JavaScriptのみならず、プログラミングの基本が身につくようになっています。

> kindle版などの電子書籍をご購入いただいた方には、この問題集のPDF版をダウンロードしていただけます（コードなどのコピー・ペーストが可能です）。解答例もウェブで公開されています（ダウンロードも可能です）。詳細は、最後にある「あとがき」を参照してください。

対象読者

この問題集の対象読者は次の方々です。

- **上記の動画を視聴中、または視聴なさった方**
  この問題集で実習をしていただくことができます。独力で解ける方もいらっしゃるでしょうが、プログラミングをゼロから単独で身につけるのは大変です。まず自力でやってみて「難しくて無理そう」と感じた場合は、下で説明する「実習編」などにご参加なさることをおすすめします
- **ほかの言語をご存じでJavaScriptを学びたい方**
  動画をご覧にならずに独力で解いていただくことも可能かと思われます。必要に応じて動画や資料をご覧ください

疑問点などがありましたらまずサポートページ（https://musha.com/sc/jsut）の資料をご覧ください。動画や資料を参考になさった上で、ご質問等がある場合はサポートページからご連絡いただくか、「実習編」あるいは「プログラミング モクモク会＋」にご参加ください[*1]。

関連講座

関連の講座については次のページをご覧ください。

- 『実習編 JavaScriptで学ぶ プログラミング入門 丸１日コース』（対面およびオンライン） —— https://musha.com/sc/jsj
  この問題集を解いていただくための講座です。ご質問等がある方、一人で解くのは大変（無理）だと思う方はご参加ください。講師（この本の著者）がていねいにご説明します
- 『プログラミング モクモク会＋』 —— https://musha.com/sc/jsmk
  上記の「実習編」に毎月ご参加いただけるサブスク（毎月定額お支払い）型のサービスです。講座にご参加いただけるほか、メールやLINE等でのご質問も受け付けます

[*1] この問題集に関するご質問は無料でお受けしております。また、この問題集以外のプログラミング等に関するご質問も、時間がかかりそうなものでな限りお受けいたします（将来的には変更する可能性がありますので、あしからずご了承ください）。

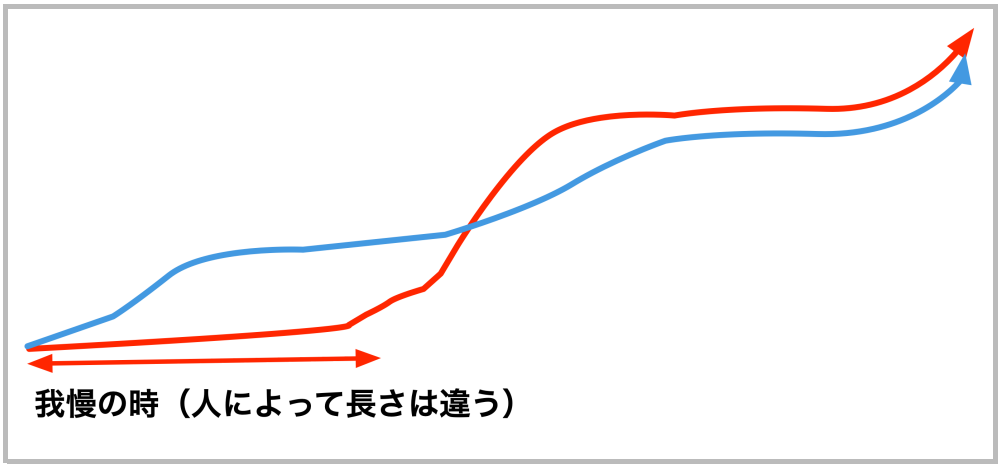
準備

次のものをご準備ください。

- パソコン —— 下記のアプリケーション（アプリ）が動作するパソコン。Windows（Win）でもMacintosh（Mac）でも構いません。また、Chrome BookやLinuxなどでも大丈夫ですが、こうした環境についての詳しい説明はしませんので、必要に応じてご自分でお調べください
- エディタ（テキストエディタ） —— 普段お使いのものがあればそれをお使いください。まだエディタを使ったことがない人はこの講座のサポートページ（https://musha.com/sc/jsut）の「エディタとブラウザ」を参考にインストールしてください
- Google Chrome（ブラウザ） —— 2020年以降に公開されたものならば大丈夫です（セキュリティを考慮すると最新バージョンの利用を推奨します）。そのほかのブラウザでも問題なく実行できますが、メニュー項目の名前などが異なる場合があります

あと、必要なのは皆さんの「忍耐力」です。「コードを直してはブラウザで確認」を何度も何度も繰り返すことになるはずです。慣れてコツがわかってしまえば、さほど苦ではなくなりますが、最初はかなり大変だと思います。

筆者はプログラミングの実力は下の図の赤い線のように向上していく人が多いのではないかと感じています。



いろいろなタイプの人がいますが、「我慢の時」を乗り越えられると一気に実力がアップするように思います。ただ、独力で「我慢の時」を乗り越えるのは大変かもしれません。「大変すぎる」と思ったら、どなたか相談できる方を見つけるとよいでしょう。そういう人が周囲に見つからないのなら、対面あるいはオンラインの「実習編」にご参加いただくと、少しは楽になるかと思います。

謝辞

この問題集は、ストアカ、Doorkeeper、connpassで公開した講座にご参加いただいた2,000人を超える皆様のフィードバックにより、長年に渡り改良されてきました。ご参加いただいた皆様およびサイト運営者の皆様に心より感謝いたします。

一部のイラストで「ねこ画伯コハクちゃん」（https://kohacu.com）の作品を使わせていただきました。ありがとうございます。

目次

第0章　初心者のためのプログラミング超入門

プログラミングとは何か。ここでザックリと説明します。公開中の動画では詳しく説明していますので、是非ご覧ください ── https://musha.com/sc/jsj

3つの側面

プログラミングをマスターするために身に着けなければない基本的な概念としてとして、次の3つがあげられます。

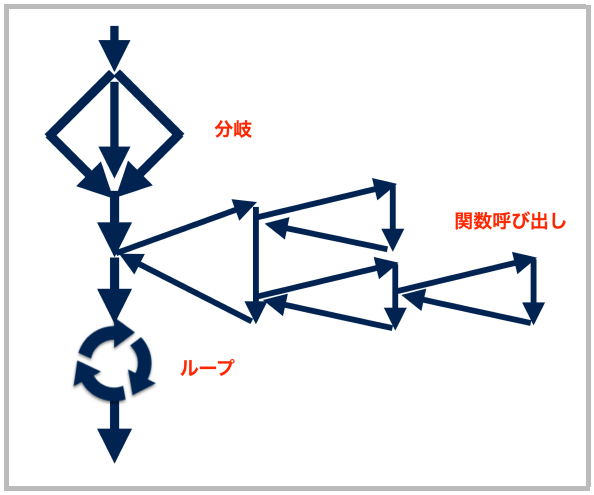
1. フロー制御 ── 計算の手順
2. データ構造 ── 情報の記憶手法
3. 演算子 ── 加減乗除や文字の連結、大小、真偽の判断などの操作

フロー制御

どのような処理をどのような順序で行うかを決めます。次の3つが基本です。この問題集の前半（第6章まで）には、この3つの概念を理解するための問題が用意されています。

- 関数呼び出し（第1章） ── 「レゴブロック」のようなもので、関数を呼び出すことでいろいろな機能を実行します。関数（の呼び出し）を組み合わせてプログラムを作っていきます
- 繰り返し（ループ。第5章） ── 人間は単純な繰り返しは嫌いですが、プログラムでは何万回でも何億回でも似たような処理を行います
- 分岐（第6章） ── 「場合分け」によって、問題を解決します

これを図にすると次のようなイメージになります。



ただ、上の図は単純化したもので、関数の中で分岐やループが使われることもありますし、分岐の中にループがあったり、逆にループの中に分岐があるといったように、さまざまに組み合わさって使われます。

データ構造

パソコンには「メモリ」があり、そこにいろいろな情報を記憶しながら、いろいろな処理を行っていきます。

- **変数**（第2章）あるいは**定数**（第4章） ── いろいろな値を記憶したり、記憶しておいた値を取り出して処理に利用
- **配列**（第10章） ── まとめてたくさんの値を記憶
- **オブジェクト**（第11章） ── 配列よりも複雑な構造を記憶

単純な変数や定数はひとつの値だけを記憶しますが、配列やオブジェクトを変数や定数に記憶することもで

きます。さらに「オブジェクトの配列」「オブジェクトを要素としてもつオブジェクト」といったものも使います。

演算子

いろいろな処理をするのに、数や文字列（文字が並んだもの）などを用いていろいろな「計算」をする必要があります。これに演算子を使います。

- 加減乗除と割算の余り（第2章） —— +、−、*、/、%
- 文字列の連結（第3章） —— +
- 比較（第6章） —— <、>、<=、>=
- 論理演算（第15章） —— &&（A**かつ**B）や||（A**あるいは**B）など

上で簡単に説明した一つひとつはそれほど難しいものではありませんが、複雑な問題を解決するためのプログラムを作成するには、これらの概念をうまく組み合わせなくてはなりません。

以下の各章で実際に問題を解くことで、各概念のイメージを掴みつつ、組み合わせ方をマスターしていきましょう。

第1章　関数の呼び出し

それでは実際に問題を解いていきましょう。次の点に注意してください。

- 問題中にある**サンプルデータやその他の資料はサポートページからダウンロードできます**（https://musha.com/sc/jsut）。サンプルデータのZIPファイルを展開すると、studentというフォルダができます。そのフォルダの中に例題用のファイル（たとえば問題1.1で使うex0101.html）が入っています。Windowsの方はstudent.zipを右クリックして、「展開」しておくことをお忘れなく。**展開しておかないと、例題がうまく動きません**
- プログラミングが初めての方は、サポートページをよくお読みの上、動画と並行して（あるいは動画を最後までご覧になってから）問題に取り組んでください
- ほかの言語をご存じの方は、単純な問題はスキップしていただいてかまいません。ただし、新しい事柄の解説が含まれている場合があるので、問題文はお読みください
- コメント（行末の「`// ...`」や「`/* ... */`」の部分）は入力する必要はありません。説明です
- コード中の「●●」や「...」の部分は考えて埋めてください（「...」は省略を意味する場合もあります）
- 解答例は「あとがき」に記載されているページにあります（ダウンロードもできます）。まずはご自分でトライしてみてから、解答例をご覧ください
- 【写経】と書いてある問題は、書いてあるコードを書き写すだけで動作します。お坊さんが書き写すことでお経を学ぶように、書き写すことでプログラムを学んでください。ただし、ただ書き写すのではなく、何をしているのか、考えながら入力して実行してください。入力したものの意味がわからない場合は動画やスライドを確認してください。お坊さんの「写経」は最初は意味がわからなくてただ写すだけでもかまわない（のかもしれない）ので、ここの「写経」とは少し意味合いが異なります
- 「バリエーション」はその上にあげた問題を少し変化させたものです。「こうしたらどうなるんだろう？」とか「こんなのもできるかな？」と自分なりのバリエーションを考えて、実際に動くものを作ってみると実力が上がるでしょう

1.1　ダイアログボックスを使って「こんにちは」と表示せよ（ダウンロードしたstudentフォルダにあるex0101.htmlを利用する）【写経】

ダウンロードしたstudentフォルダにあるex0101.htmlを利用する。

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>1-1 ダイアログボックスの表示</title>
</head>
<body style="margin: 5rem; font-size: xx-large;">
  <p>
    ダイアログボックスに「こんにちは」と表示しました。
  </p>
<script type="text/javascript">
  "use strict"; // これがあるとエラーが見つけやすくなる
  alert("こんにちは");
</script>
</body>
</html>
```

■メモ

実践では（製品として公開するシステムをほかの人と共同で作成したりする場合には）通常、HTMLのコードとJavaScriptのコードは別のファイルに書きます。そのほうが、共同作業がしやすいし、（ファイルが大きくなると特に）管理が楽です。上のコードは、たとえば次のように2つのファイルに書くことができます。

```
// ファイル ex0101tip.html
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>1-1 ダイアログボックスの表示（別ファイル）</title>
</head>
<body style="margin: 5rem; font-size: xx-large;">
  <p>
    ダイアログボックスに「こんにちは」と表示しました。
  </p>
<script src="ex0101-tip.js" type="text/javascript"> <!-- src= でJSファイルを指定 -->
</script>
</body>
</html>
```

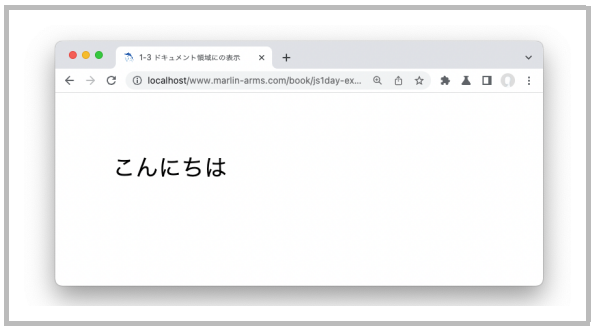
```
// ファイル ex0101tip.js
"use strict";
alert("こんにちは");
```

この問題集の例は短いものが多いのでJavaScriptのコードもHTMLファイルに書いている場合が多いですが、実践では（ほとんどの場合）JavaScriptのコードは別ファイルに書くようにしましょう。

1.2　alertの引数（"こんにちは"）をいろいろ変えてみて、違うメッセージが表示されることを確認せよ

```
alert("●●"); // ●● を適当に変える
```

1.3　alert(...)の代わりにdocument.write(...)を使い、ダイアログボックスではなくブラウザのウィンドウ内（「ドキュメント領域」と呼ぶことにする）に表示されることを確認せよ【写経】




```
document.write("こんにちは");
```

■メモ

実は、実践でdocument.writeを用いることは（ほとんど）ありません。この問題集ではしばらくdocument.writeを用います。HTMLを知っている人にとって、とてもわかりやすいと思うからです。実践では、document.writeを使っているところを、後で紹介する関数などに置き換えます。

なぜ実践でdocument.writeを使わないかについては、第8章の最後のコラムで説明します。

1.4 document.writeを使ってstudentフォルダの中のpicturesフォルダにある画像を表示せよ【写経】

```
document.write(`<img src='pictures/picture000.jpg'>`); //「`」の代わりに「"」でもOK
//  「`」はバッククオート。多くのキーボードでは「Shift+@」。英語キーボードでは左上にある
```

- 画像があることを確認
- うまくいかない場合はHTML（JavaScript）のファイルと画像ファイルとの位置関係を確認すること。studentフォルダの中にHTMLファイルを置けば上のコードで動くはず

■警告

画像がうまく表示されないときは次をチェック。

- ファイル名が間違っていないか、スペルミスはないか
  - フォルダ名はpictureではなくpictures（複数の画像が入っているので）
  - ファイル名はpicture000.jpg（ひとつのファイルについて言っているのでpicturesではない）
- 一部のエディタでは、HTMLファイルのある位置をトップとしてウェブサーバを起動してコードを実行するので、次のようなコードを書くと画像にアクセスできないことがあるので注意。その場合、画像のフォルダをコードのあるフォルダと同じ位置（あるいは下）に置く（上のコードはそうなっている）

```
document.write(`<img src='../pictures/picture000.jpg'>`); // 動かないことがある！
// 上のフォルダ（ディレクトリ）から別の場所にあるフォルダを参照するのは、
// ウェブサーバを起動するエディタではうまく行かないことがある
```

バリエーション

- 連続して書くことで、複数の画像が表示されることを確認せよ

```
document.write(`<img src='pictures/picture000.jpg'>`);
document.write(`<img src='pictures/picture001.jpg'>`);
document.write(`<img src='pictures/picture002.jpg'>`);
```

第2章　演算子（加減乗除）

2.1　次の計算の結果を予想し、結果を出力して比較せよ（あとで復習できるように、前の章の例題とは別のファイル名で保存することを推奨。たとえばex0201.html）

```
let x = 320; // xという名前の変数（情報の記憶場所）に320を記憶する
             // 「=」の意味は下記参照。「;」は「文」の区切り。行の最後ならば省略できる
x = x - 20;  // xに記憶されている値からから20を引いて、再度xに記憶する
x += 33; //  「x = x+33;」と同じ。 xに33を足してそれをx記憶する
x -= 21; //  「x = x-21;」と同じ。 xから21を引いて、それをx記憶する
x++; //  「x++」は　「x = x+1」と同じで、xに1を足してそれをxに記憶する
     // 1を足す操作はよく使われるので特別な演算子(オペレータ)++が用意されている
document.write(x); // ドキュメント領域にxの値を出力する
```

■メモ

「x = 320」の「=」は数学で使う「イコール」とは意味が違う。どちらかというと次のようなイメージで、xという名前がつけられた記憶域に320という値を記憶する、記憶域（メモリ）内の状態の変化を引き起こす。

```
「x ← 320」
```

2.2　次の計算の結果を予想し、結果を出力して比較せよ

```
let x = -3;
x -= 20; //  「x = x-20;」自分（x）から20を引いて、それを自分（x）に記憶する
x += 3;
x--; //「x--」は　「x = x-1」と同じで、xから1を引いて、それををxに記憶する
     // 1を引く操作はよく使われるので特別な演算子(オペレータ)（--）が用意されている
document.write(x);
```

バリエーション

- +, -, *, /, %（余り）を使っていろいろな計算をしてみよ

2.3　今日のラッキーナンバー(0〜9までのいずれか）を表示するプログラムを作成せよ。0〜9までの数字はランダムに出現するものとする【写経】

```
// Math.random() -- 0以上1未満の小数をランダムに返す
// Math.floor(x) -- xを整数に切り下げる
// document.write("xxx") -- xxx をHTMLコードとして出力
//
let x = Math.random();  // 0 ≦ x < 1
x *= 10; //  「x = x*10」 と同じ。自分を10倍したものを、自分（x）に記憶。 0 ≦ 10x < 10 になる
x = Math.floor(x);
document.write(x);
```

2.4　今日のラッキーナンバー(1〜10までのいずれか）を表示するプログラムを作成せよ。1〜10までの数字はランダムに出現するものとする

```
// まず0から9を出し
let x = Math.random();// 0 ≦ x < 1
x *= 10;              // 0 ≦ 10x < 10
x = Math.floor(x);    // 0以上の整数 0 ≦ 10x ≦ 9（切り下げて整数になる）
 【ここで1を追加する操作を行う】// ←考えてください
document.write(x);
```

2.5　上の2-1の各文の後ろにconsole.log()やalert()を入れて、途中経過を表示しながら、xの値の変化を追ってみよ

```
let x = 300;
x += 33;
x -= 21;
x++;
```

↓

```
let x = 300;
console.log(x);
x += 33;
console.log(x);
x -= 21;
console.log(x);
x++;
console.log(x);
```

- console.log()の出力結果が表示される「console（コンソール）」を表示するには次のいずれかを実行（次はChromeの場合。他のブラウザにも同じような項目がある）
  - ドキュメント領域で右クリック→［検証］→［Console（コンソール）］。なお、Macで右クリックを設定してない場合は「control+クリック→［検証］→［コンソール］
  - ショートカットキー ―― WinではCtrl+Shift+J、Macでは⌘+option+J
  - メニューから［表示］→［開発/管理］→［JavaScript コンソール］

第3章　文字列の連結

3.1　次のコードを読み、何が出力されるか予想してから、入力して実行せよ

```
let a = "武田"; // '武田' でも `武田` でもOK
let b = "信玄";
let c = a + b;
document.write(c)
```

文字列の連結にも「+」を使う。上のコードでは「武田信玄」がドキュメント領域に書かれることになる。

3.2　次の文字列処理の結果xの値がどうなるかを予想し、結果を出力して予想と比較せよ

```
let x = "JavaScript";  // let x = 'JavaScript';    でもOK
x += "入門"; // xの後ろに"入門"が追加される
let y = "実習編もどうぞ --";
x = y + x;
x = "「" + x + "」"; //数字と文字列、文字列と文字列をつなげるには「+」
// なお、第4章でやるバッククオート（`...`）も使える
document.write(x);
```

第4章　テンプレートリテラル

4.1　次のバッククオート（`）を使った文字列処理の結果xの値がどうなるかを予想し、結果を出力して比較せよ

```
1: let x = "JavaScript";
2: let y = "入門";
3: x += y;
4: x = `『${x}』`; //  「`」はバッククオート（多くのキーボードでは「shift+@」で入力）
5: //   `...`の中では、${...}で囲むと変数を使ったり、計算式を使ったりできる
6: document.write(x);
```

上のリストの4行目のバッククオートで囲まれた文字列（`『${x}』`）の中には変数が含まれている。このような表現を「テンプレートリテラル」という。**${...}**の「...」の部分にある変数はその値が計算されて文字列に置き換わる。テンプレートリテラルは「可変部分付き文字列」である。

4.2　次の計算の値がどうなるかを予想し、結果を出力して予想と比較せよ

```
const a = 2; //  「let a = 2」でもよいが、値が変わらない場合はconst（定数）を使うほうが、なおよい
const b = 3;
document.write(`a*b=${a*b}<br>`);  // `...` の中で計算をしている
// constはletとよく似ているが、それ以後に変更しない場合に用いる
// letを用いても結果は同じだが、この値は変わらないことを読む人（将来の自分も含む）に意識してもらえる
```

■追加情報

constはletとほとんど同じ働きをするが、constで宣言しておくと、その変数の値が変わらないことが明確になる。

なお、後半のほうで登場するオブジェクトをconstで宣言した場合、そのオブジェクトの構成要素（プロパティ）の値は変更できてしまうので、必ずしもこれが当てはまらない。このため、constは単純な変数について使う場合が多い。

4.3　次のコードを実行して画像が（300ピクセル［px］の幅で）表示されることを確認せよ【写経】

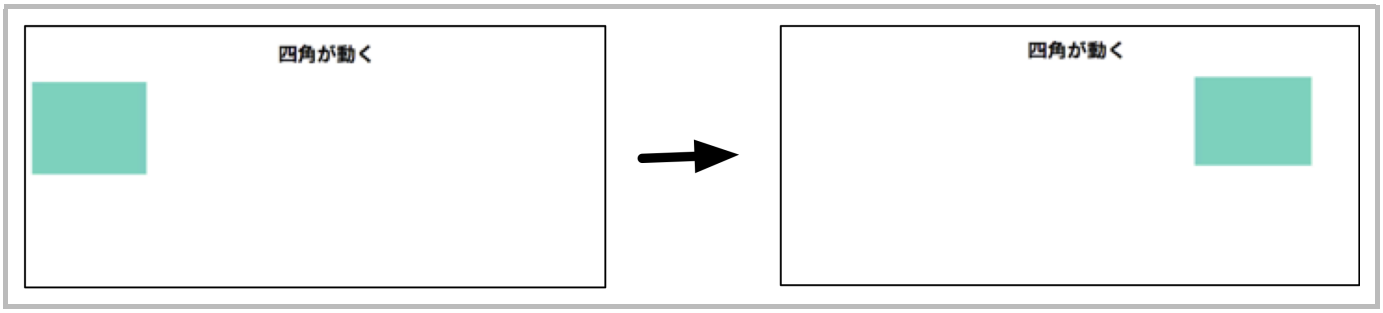
```
const ファイル名 = "pictures/picture000.jpg"; // 変わらない値はletよりconstを使うほうがよい
const 画像タグ = `<img src="${ファイル名}" style="width: 300px;">`;
document.write(画像タグ);
```

バリエーション

- 上のコードのwidthの値をいろいろの変更し、画像の大きさを変えてみよ。たとえば、200px, 50%, 25%, 20rem, ...など（remは文字と同じ幅を表す。20remは20文字分）

第8章　アニメーション

8.1　画面上に長方形を表示し、その長方形を左から右に動かすプログラムを作れ



例（方法はいろいろあるが一例。animation.htmlにあり。「●●」を要変更）

- CSSで背景色を使って四角を書く
- マージンをJavaScriptで変化させる

コード

```
...
<body>
<div id="rectangle" style="background-color: #6DBBA1;
         width: 200px; height: 180px;">
</div>

<script>
"use strict";
let カウンタ = 0; // 左端からの距離（px（ピクセル） 単位）
const タイマーID = setInterval(四角を動かす, 10); // 時間は適当な長さに。1/1000秒単位

// setIntervalで、連続して「四角を動かす」を呼び出す
function 四角を動かす() {
  カウンタ++;
  const 四角 = document.getElementById("rectangle");
  // <div id="rectangle">...</div>の下にJavaScriptのコードを書くこと！
  // "rectangle"のIDをもつ部分が、この上にないとそのオブジェクトを取得できない
  四角.style.marginLeft = `${カウンタ}px`;
  // CSSのmargin-left（左端から四角までの距離）を指定
  if (●● < カウンタ) { // 終了させる　←★★要変更★★ 数値を入れる
    clearInterval(タイマーID);
  }
}
</script>
</body>
</html>
```

8.2　前の問題の長方形を5倍の速度で動かせ。

- たとえば、マージンの指定を5倍にする

8.3　画面上に長方形を表示し、その長方形を左から右、右から左と交互に動かすプログラムを作れ

方法（一例）

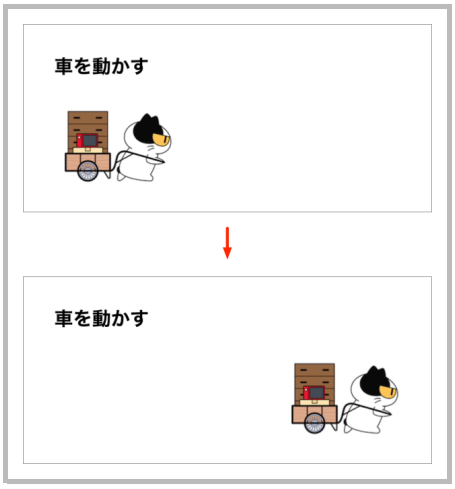
- 右側に来たら、方向を変える　→ marginの値を減らしていく

コード例

```
"use strict";
const 右端 = 800; // px単位
const 倍率 = 5;
let カウンタ = 0;
let 方向 = 1;  // 1が右方向 -1が左方向
const タイマーID = setInterval(四角を動かす, 10);
```

```
function 四角を動かす() {
  カウンタ++;
  let マージン;
  if (0 < 方向) { // 右方向の場合
    マージン = カウンタ*倍率;
  }
  else { // 左方向の場合
    マージン = 右端 - カウンタ*倍率;
  }
  const 四角 = document.getElementById("rectangle");
  四角.style.marginLeft = `${マージン}px`;
  if (右端 < カウンタ*倍率) {
    カウンタ = 0;
    方向 *= -1; // 「方向 = 方向*(-1)」と同じ意味。この書き方のほうが簡潔
    // 方向を反対にする  1*(-1) => -1;   (-1)*(-1) => 1;
  }
}
```

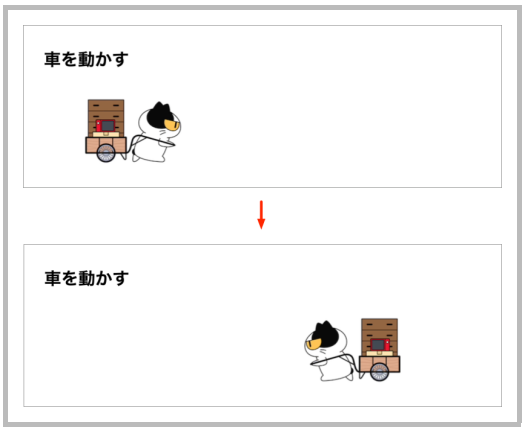
8.4　画面上に車の画像（pictures/car1.png）を表示し、その車を左から右に動かすプログラムを作れ



- 背景ではなく、画像のマージンを変化させる
- たとえば、次のようなタグを用いて画像を表示して、 ... のマージンを増やしたり減らしたりすればよい（このようにすれば、四角を動かす問題と同じ手法が使える）

```
  <div id="car">
    <img src="pictures/car1.png">
  </div>
```

8.5　車を左から右、右から左と交互に動かすプログラムを作れ（pictures/car1.pngとpictures/car2.pngの画像を使う）



8.6　上の問題で、ウィンドウの右端より少し左で折り返すようにせよ

- window.innerWidth でウィンドウの幅（ピクセル単位）がわかる
- これがピクセル単位なので、移動する単位もピクセルにしたほうがよい

8.7　上の問題で、ウィンドウの大きさを変えたときにも右端より少し左で折り返すようにせよ

- ウィンドウの大きさを変えるとresizeイベントが発生するので、resizeイベントを処理すればよい

```
window.addEventListener("resize", () => {
  // ウィンドウのサイズが変わったときの処理をここに書く
});
```

8.8　画面上に長方形を表示し、その長方形を左から右に動かすプログラムを作れ。長方形が移動するとき、色がランダムに変わるものとする（★ちょっと難問★）

```
/* 以下のコードは参考。一部分のみ */
"use strict";
// 色を変える
let 四角 = document.getElementById("rectangle");
四角.style.backgroundColor = "#BA03EE";

let 色 = ランダムな整数を生成(0, 15)
let 色の文字列 = 色.toString(16); //文字列を16進数に
//「ランダムな整数を生成」はjs1daylib.jsの中にあり
```

document.writeについて

1. 実行速度

入出力操作（画面表示、ファイル読み込みなど）は実行に時間がかかることが多い。したがって、document.writeもその都度呼び出すのではなく、表示したいものをまとめておいて一度に出力するほうが効率がよい。

この講座では初心者にもわかりやすくするために、特に最初のほうではその都度document.writeを呼ぶ例を示したが、ある程度以上の規模のプログラムを作るときは実行速度も考慮する必要がある。たとえば次のように書き出すコードを変数に追加する形で記憶しておいて、最後にまとめて出力するようにしたほうが実行が速くなる。

```
let x = "....";
x += "...";
...
x += "...";
document.write(x);
```

2. ブラウザによる非同期処理の挙動の違い

document.writeの非同期の処理（タイマーやネットワーク経由の処理など即座に実行されない処理）の動作は（残念ながら）ブラウザによって異なる。このため非同期の処理の場合document.writeは使わないほうがよい。

非同期の処理などを行う場合は、たとえば次のように ... とdocument.getElementByIdの組み合わせなどを使って、その部分のinnerHTMLに代入してしまえばdocument.writeを使わずに済む。

```
【htmlファイル】
...
<div id="yyy">
  <!-- ここに書き込む -->
```

```
</div>
...
<script src="xxx.js" type="text/javascript"></script>

【JSファイル】
"use strict";
let x = ...;
x += ...;
..
obj = document.getElementById("yyy"); // ID "yyy"のオブジェクトを得る
obj.innerHTML = x; //  「ここに書き込む」のところに書き込む
```

3. document.write()の評判が悪い

次のようなスクリプトを書くと、document.write() によって外部のスクリプトを動的に挿入することができるが、このような処理をするとスクリプトを読み込む間、表示用の処理ができないので、コンテンツの表示がかなり遅れることがある。

```
document.write('<script src="https://www.xxx.com/yyy.js"></script>');
```

このような「動的なスクリプトの挿入」など遅延の原因になる処理を行わなければdocument.writeを使っても明らかな不具合は生じないと思われる。ただし、一概にdocument.writeを「使うな」あるいは「非推奨」とする傾向がある。たとえば、Google Chromeでは「使うな（Avoid using document.write)」といったメッセージが表示されるようになった（おそらくこうしたほうがChrome側の処理が簡単だからだと思われる。詳細はhttps://developers.google.com/web/updates/2016/08/removing-document-write）。

上の2.にあげたような方法を使えばdocument.writeを使わなくても済むので、将来、ウェブに公開するようなスクリプトを書く場合にはdocument.writeは使わないようにしたほうがよい。練習に用いたり、自分用のアプリを作るのには用いてもかまわない。

いろいろ理由をあげたが、「プロなら本番環境ではdocument.writeは非効率だから使うな。ほかに方法がある」ということだ。if文やfor文の機能を覚えたりするために練習用に使うのはかまわない（ブラウザでサポートされなくなれば話は別だが）。HTMLを知っている人にとって、こんなにわかりやすい関数はないのだから。

●著者紹介

武舎 広幸 （むしゃ ひろゆき）

国際基督教大学、山梨大学大学院、リソースシェアリング株式会社、オハイオ州立大学大学院、カーネギーメロン大学機械翻訳センター客員研究員等を経て、東京工業大学大学院博士後期課程修了。マーリンアームズ株式会社（https://www.marlin-arms.co.jp/）代表取締役。主に自然言語処理関連ソフトウェアの開発、コンピュータや自然科学関連の翻訳、プログラミング講座や辞書サイト（https://www.dictjuggler.net/）の運営などを手がける。訳書に『Symantec C++トレーニングブック』（翔泳社）、『HTML入門 第2版』『Java言語入門』『Perl入門 第2版』（以上プレンティスホール）、『Goプログラミング実践入門』『Python基礎&実践プログラミング』（以上インプレス）『初めてのJavaScript 第3版』『初めてのGo言語』『インタフェースデザインの心理学 第2版』『AIの心理学』（以上オライリー・ジャパン）など多数がある。https://www.musha.com/にウェブページ。

JavaScriptで学ぶプログラミング入門 丸一日コース 問題集 第4版

2022年12月15日　初版第1刷発行
2023年 1月15日　第2版第1刷発行
2023年 2月24日　第3版第1刷発行
2023年 4月10日　第4版第1刷発行

著　者　武舎（むしゃ） 広幸（ひろゆき）
発行人　武舎 広幸
発行所　マーリンアームズ株式会社 https://www.marlin-arms.co.jp