

『実践JavaScript!』 講座用練習問題集

講座での実習用に、教科書から第12章までの練習問題を抜き出したものです。

- 一部表現を変えてある箇所がありますが、問題番号（「問題1-1」など）は教科書と同じになっています
- 教科書の文章に説明を追加したり、メモ的な説明を省略しているところがあります。省略しているところには、その旨が書いてありますので（後ほど）教科書で確認してください
- 第7章までの内容は入門編（やビデオ）で説明しています。第8章以降の問題を解く際には、まず教科書本文を読んでから取り組んでください（問題11-4から問題11-9を除く）
- 不明点がある場合は、スライドのコピーや教科書本文を参照してください。その時、教科書の索引などもご利用ください
- 少し調べてもわからなかったら、早めに講師にお尋ねください（自習なさっている場合は、通信機能やメールなどでお尋ねいただいても結構ですし、まとめておいて講座のときに聞いていただいても結構です）
- ダウンロードしていただいたjsdataフォルダにあるindex.htmlをブラウザで表示すると、**練習問題（や本文中の例題）の一覧を表示し、簡単に実行できます。是非ご利用ください**（WindowsのエクスプローラやmacOSのファインダでダブルクリックしたりして開いてください。VS Codeから実行すると、閉じないと他のファイルを実行できなくなりますので）。

●初めて実習編にご出席中の方

- 午前の実習は、基本的には最初からやってみてください。他のプログラミング言語をよくご存じの方は「これは大丈夫」と思う問題は飛ばしていただいても結構です
- 午後の実習は次の順番でやってみてください。簡単なアニメーションが作れます
 - 問題6-1～6-4 —— タイマーの使い方
 - 問題11-4～11-9 —— タイマーを使ったアニメーション。example/ch1104.htmlをベースにしてください
 - 問題11-9まで終わったら、午前実習の続きをやってください

●実習編が2回目の方（あるいは実習編のあとで入門編をお受けになっている方）

—— 最初の問題から始めて、「前回やったので大丈夫」と思うものは飛ばして進んでください

●実習編が3回目以上の方 —— 前回の続きからやってください

第1章 1.11 練習問題

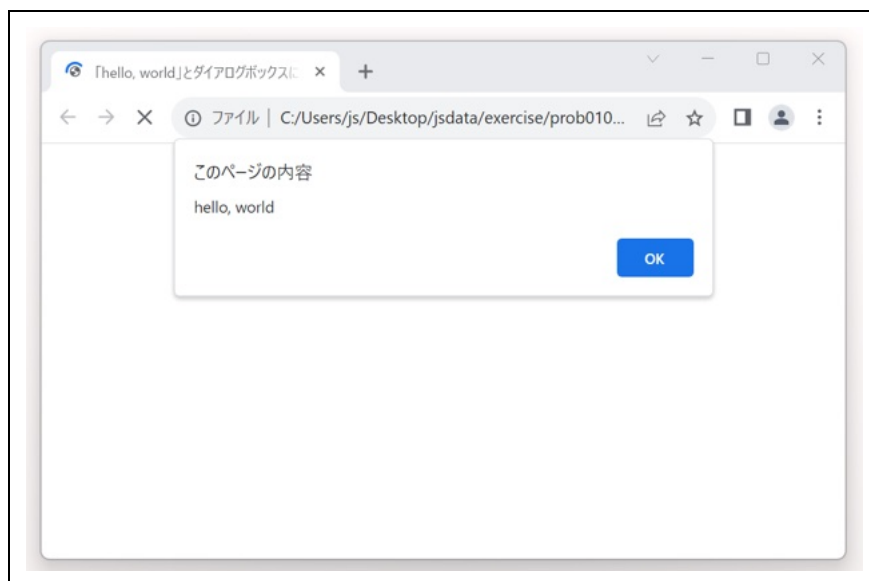
実際にプログラムを作って実行してみましょう。一人でやってもまごつかないよう、とても単純な問題からやっていきますので、安心してついてきてください（最初の数章は比較的単純な問題が多いですが、徐々に難しめの問題が入ってきます）。

■Note

【写経】と書いてある問題は、その問題に書いてあるコード(プログラム)をそのまま入力して実行すればできる問題です(入力も不要なものもあります)。お坊さんが書き写すことでお経を学ぶように、プログラムを書き写すことで学んでください。

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、教科書の本文やスライドを参照してください。

問題1-1 ex0101.htmlをブラウザで開いて、ダイアログボックスに「hello, world」と表示されることを確認せよ【写経】

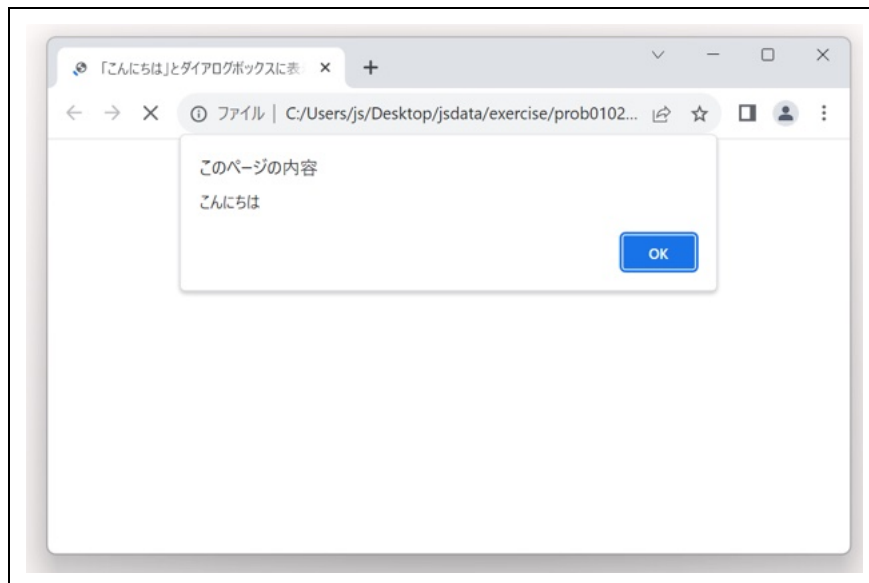


すでに試した方は、この問題をスキップしていただいて結構です。まだ試していない人は、jsdataフォルダにあるexampleフォルダの中にあるex0101.htmlを選択して、[実行]→[デバッグなしで実行]→[Web アプリ(Chrome)]の順に選択して、Chromeでダイアログボックスを表示してみてください。なお、「○○フォルダにある～○○フォルダにある...」と何度も繰り返すのは面倒ですので、以降「jsdata/example/ex0101.html」のように、フォルダ名を「/」で区切って書くことにします。一般に広く使われている表記法です^{*11}。

[*11] Windowsでは「/」の代わりに「\」あるいは「\\」が使われていました。ただ、最近ではWindowsでも「/」が使われる場合が増えてきています。この本では「/」で統一します。


ブラウザにダイアログボックスが表示されたら、[OK]をクリックして、ダイアログボックスを閉じましょう。続いて、ブラウザのウィンドウも閉じてください。

問題1-2 ダイアログボックスを使って「こんにちは」と表示せよ



同じファイルex0101.htmlをVS Codeで開いて、「hello, world」のところを「こんにちは」に変更して、ブラウザで表示するのが簡単です。

よくわからない方は次の手順に従ってみてください。

1. VS Codeのエクスプローラーでex0101.htmlを選択する
2. 右側のエディタペインに表示された`alert("hello, world")` を `alert("こんにちは")` に変更する
3. ファイルを保存する (^+S押すのが簡単。Controlキーを押しながらSのキーを押す [ ⌘ +S])
4. メニューから [実行] → [デバッグなしで実行] を選択 (先ほど同じファイルに対してChromeで実行することを選択してあるので、再度Chromeを選択する必要はありません)

なお、解答例のファイルも、jsdataフォルダのexerciseというフォルダに入っています(「1.12 練習問題の解答例」に詳しい説明があります)。

間違えて全角の引用符や全角の括弧を用いたりすると、VS Codeがエラーを見つけてくれて指摘してくれます。たとえば、図1.17では、「(」を全角で入力しています。そうするとJavaScriptの処理系は、この行全体をうまく解釈できないため、怪しい部分の文字の下に赤い波線を引いてくれます。このように赤い波線が引かれている部分がいったら、修正する必要があります。

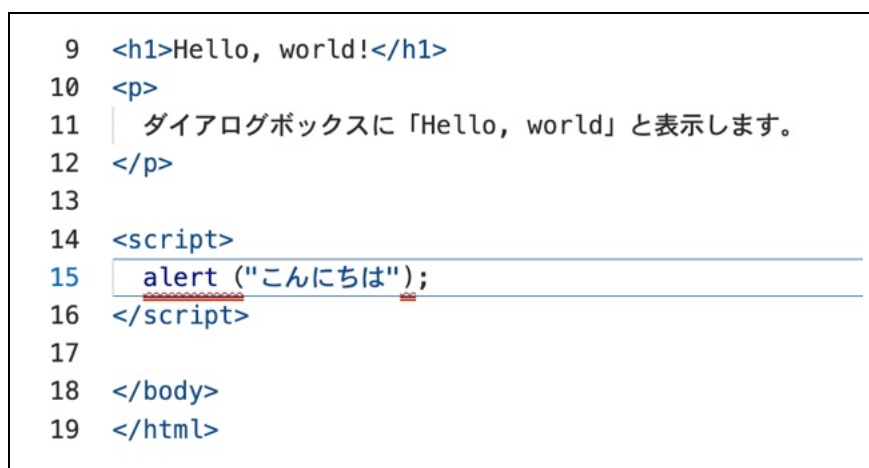


図1.17: VS Codeが構文エラーを見つけると赤い波線を引いてくれる

■注意

VS Codeからブラウザが起動できないときは

VS Codeでブラウザがうまく起動されなくなることがあります。

その場合はサイドバーに表示されるエクスプローラーで、`.vscode`というフォルダの下に`launch.json`というファイルがあったら、それを削除してから再度実行してみてください(右クリックして[削除]を選択^{*12})。

それでもうまくいかない場合、(このフォルダの下に自分で何も作ってない場合は)`.vscode`というフォルダごと削除してみてください。

なお、既にファイルを実行していると、別のファイルの実行はできません。そのファイルの実行を終了してから起動し直してください。

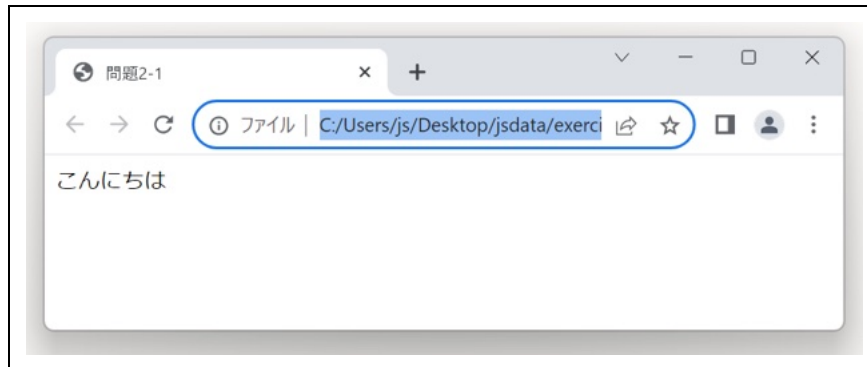
[*12] **mac** macOSではcontrolキーを押しながらクリックして[削除]を選択してください。なお、macOSの[システム設定]で[マウス]→[副ボタンのクリック]で[右側をクリック]を指定しておくと、controlキーを押さなくても、マウスの右側(あるいは右ボタン)を押すだけで「右クリック」できます。

1.12 練習問題の解答例

練習問題の解答例は、ダウンロードしたフォルダの`jsdata/exercise`に入っています。たとえば、最初の問題1-1(ダイアログボックスに「hello, world」を表示)の解答例は`jsdata/exercise/prob0101.html`に入っています。これからの問題についても同じ形式のファイル名に解答例が入っています。たとえば、第5章の「問題5-2」の解答例は`exercise/prob0502.html`にあります。

VS Codeのエクスプローラーで`example`フォルダの中身が表示されていたら、`example`を一度クリックして閉じて、その下にある`exercise`をクリックすると、解答例のファイル一覧が表示されます。問題1-2[ダイアログボックスを使って「こんにちは」と表示せよ]の解答例ならば`prob0102.html`をクリックすると右側のエディタペインにファイルの内容(ソースコード)が表示されます。メニューから[実行]→[デバッグなしで実行]を選択することで実行できます。

問題2-1 ドキュメント領域に「こんにちは」と表示せよ【写経*9】



example/ex0101.htmlなどをVS Codeで開いて[ファイル]メニューの[名前をつけて保存...]を選んで、**exercise**フォルダの下に、**prob0201me.html**などといった名前で作成してから、コードを変更してください。解答例が**prob0201.html**などといった名前なので、**prob0201me.html**などと少し変えておくと、簡単に比較できます*10。

ファイル全体は次のようになります。

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>問題2-1</title>
</head>
<body>

<script type="text/javascript">
  document.write("こんにちは");
</script>
</body>
</html>
```

prob0201me.htmlを実行するとブラウザのドキュメント領域に「こんにちは」と表示されることを確認してください。

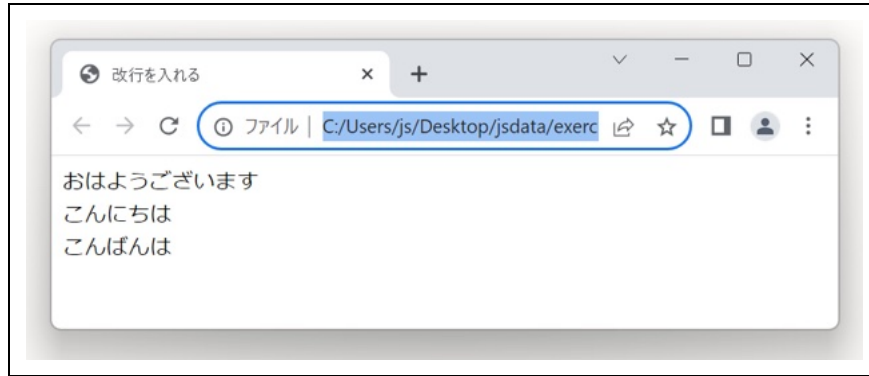
[*9] 第1章でも説明しましたが、【写経】と書いてある問題は、その問題に書いてあるコード(プログラム)をそのまま入力して実行すればできる問題です(入力も不要なものもあります)。意味を考えながら入力して実行してみてください。

[*10] 他の場所に保存してもよいですが、解答例と自分の解答を区別できるように(少し)名前を変えておいたほうが、あとで混乱しなくて済むでしょう。練習問題用に別のフォルダを作る場合は、画像や動画のフォルダ(**pictures**と**movies**)を**exercise**フォルダからコピーしておいてください。後ろの章の練習問題で利用します。

筆者の解答例が**exercise/prob0201.html**に入っています。

問題2-1の解答例ならばVS Codeのエクスプローラーで**prob0201.html**をクリックすると右側のエディタペインにファイルの内容(ソースコード)が表示されます。メニューから[実行]→[デバッグなしで実行]を選択することで実行できます。

問題2-2 下の図のようにドキュメント領域に「おはようございます」「こんにちは」「こんばんは」を1行ずつ表示せよ（間に「改行」を入れる）【写経】



【解答例】

```
<!DOCTYPE html>
<html>
...   <途中のタグは省略。以下の説明でも同様>
<script type="text/javascript">
  document.write("おはようございます<br>");
  document.write("こんにちは<br>");
  document.write("こんばんは");
</script>
...   <このあとに続くタグも省略>。
```

■HTMLメモ

ドキュメント領域の文章に「改行」を入れるにはHTMLの**br**タグを使う必要があります。次のようにすることで、「おはようございます」「こんにちは」「こんばんは」がそれぞれ別の行に表示されます。

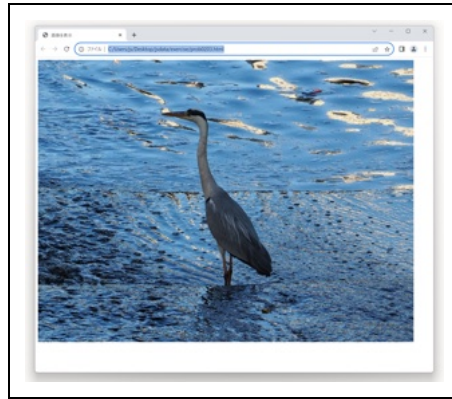
```
おはようございます<br>
こんにちは<br>
こんばんは
```

一方、次のように**br**タグを書かないと、改行はされず1行に表示されてしまいます(**prob0202b.html**)。

```
おはようございます
こんにちは
こんばんは
```

おはようございます こんにちは こんばんは

問題2-3 `document.write`を使って**pictures**フォルダにある画像**picture001.jpg**を表示せよ【写経】



```
document.write('<img src='pictures/picture001.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+@」。英語配列では左上にある
```

- 画像があることを確認してください
- 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してください。
jsdata/exerciseフォルダにHTMLファイルを置けば上のコードで動くはず

■Note

画像がうまく表示されないときは次をチェックしてみてください。

- 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）
 - ファイル名は**picture001.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）

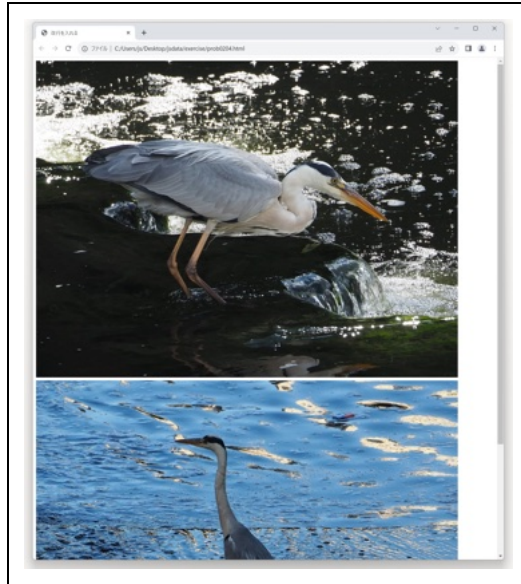
どうしてもうまく行かない場合は講師にお尋ねください（自習なさっている場合は「棚上げ」して先に進み、あとで戻ってやってみてください）。

あるいは、第8章のデバッグに関する説明を読んでみてから、戻ってトライしてみても結構です。

この本のまえがき（「イントロ —— JavaScriptと英語を比較する」）にも書きましたが、重要な概念は繰り返し登場するので、そのたびに復習していえば、徐々に身につきます。少しぐらいうまくいかないことがあっても、先に進んでみましょう。

問題2-4 `document.write`を使って**pictures**フォルダにある2つの画像**picture000.jpg**と**picture001.jpg**を表示せよ【写経】

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

バリエーション

- 表示する画像の数を、3枚、4枚、...と増やしてみよ

```
// 4枚の例
document.write('<img src='pictures/picture000.jpg'>');
document.write('<img src='pictures/picture001.jpg'>');
document.write('<img src='pictures/picture002.jpg'>');
document.write('<img src='pictures/picture003.jpg'>');
```

■Note

「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、作ってみると実力が上がるはずです。

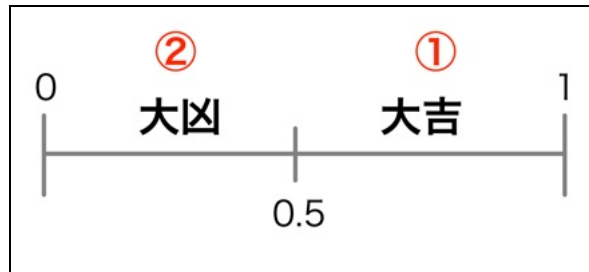
講座受講中の方は、バリエーションの問題をやってみてください。

第3章 3.15 練習問題

分岐の問題のほかに算術演算子(加減乗除)に関する問題も含まれています。

問題3-1 大吉か大凶が同じ割合でランダムに表示されるおみくじのプログラムを作れ【写経】

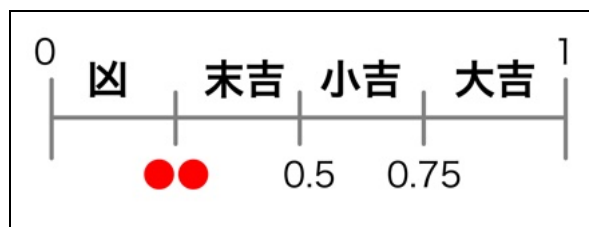
```
let x = Math.random(); // 0以上1未満の小数を返す
if (0.5 <= x) {
  document.write("大吉"); // ①
}
else {
  document.write("大凶"); // ②
}
```

教科書の問題3-1にある補足説明(p. 65～)はここでは省略します。**お時間があるときにご確認ください。**

問題3-2 大吉、小吉、末吉、凶が同じ割合で出るおみくじのプログラムを作れ

```
let x = Math.random(); // 0以上1未満の小数を返す
if (0.75 <= x) { // 4つを同じ割合で出すようにするので4分割する
  x = "大吉"; // おみくじの文字列をxに記憶しておく
}
else if (0.5 <= x) {
  x = "小吉";
}
else if (●●) { // ●●を変える
  x = "末吉";
}
else {
  x = "凶";
}
document.write(x); // 上でおみくじの文字列を作って、ここで書き出す
```



問題3-3 今日の運勢（大吉、中吉、小吉、末吉、凶、大凶のいずれか）を次の割合で表示するプログラムを作成せよ

- 「大凶」 のでる確率10%
- 「凶」 のでる確率20%
- 「末吉」 のでる確率20%
- 「小吉」 のでる確率20%
- 「中吉」 のでる確率20%
- 「大吉」 のでる確率10%

問題3-4 前の問題と同じ割合で今日の運勢（大吉、中吉、小吉、末吉、凶、大凶のいずれか）を表示するプログラムを作成せよ。このとき、`pictures`フォルダにある該当する画像（`omikuji_suekichi.png`など）を表示せよ



問題3-5 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に-3を代入する（xに-3を記憶する。「xを-3で初期化する」）
x -= 20; // 「x = x-20」と同じ。 xから20を引いてxに記憶
x += 3; // 「x = x+3」と同じ。 xに3を足してxに記憶
x--; // 「x--」は 「x = x-1」と同じで、xから1を引いて、それをxに記憶する
document.write(x); // xの値をドキュメント領域に表示する（書き出す）
```

問題3-6 次の計算の結果を予想し、結果を出力して比較せよ

```
let x = 300; // xを300で初期化する
x += 33; // 「x = x+33」と同じ。 xに33を足してそれをx記憶
x -= 21; // 「x = x-21」と同じ。 xから21を引いて、それをx記憶
x++; // 「x++」は 「x = x+1」と同じで、xに1を足してそれをxに記憶
document.write(x)
```

問題3-7 次の図は変数`x`の値の変化を順を追って表示したものである。「`x -= 2;`」以降の空欄を埋めよ

	x
let x = 4;	4
x += 5;	9
x -= 2;	
x++;	
x *= 2;	
document.write(x);	

問題3-8 同じように「`x = y + 5;`」以降の空欄を埋めよ

	x	y
<code>let x = 4;</code>	4	
<code>let y = 10;</code>	4	10
<code>x = y + 5;</code>		
<code>y -= x;</code>		
<code>y--;</code>		
<code>document.write(y);</code>		

問題3-9 上の問題の各文の後ろに`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;
alert(x); // 300が表示されることを確認
x += 33;
alert(x); // 同様に値を確認
x -= 21;
alert(x); // 同様に値を確認
x++;
alert(x); // 同様に値を確認。
document.write(x);
```

.....

`alert`を使うと途中経過を確認することができます。ただしオススメの方法ではありません。

次章以降(とくに第8章)で、もう少しエラーの修正(「デバッグ」)に便利な関数`console.log`やデバッガを使う方法を紹介しましょう。

問題3-10 次のコードを読み、何が出来されるか予想してから、入力して実行せよ

```
1 let a = "武田"; // '武田'でもOK。
2 let b = "謙信"; // '謙信'でもOK。上で"武田"としたら"謙信"と同じ形式で揃えるほうがよい
3 let c = a + b; // 文字列の連結にも「+」を使う。
4 document.write(c)
```

1行目(および2行目)はバッククォートを使って`武田`などとしてもでもOKですが、バッククォートは次章で説明する「テンプレートリテラル」で使われるので、このような単純な文字列には「`"`」か「`'`」を使うのが一般的です。

問題3-11 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 = "初めての";
x += "JavaScript"; // xの後ろに「JavaScript」が追加される
let y = "この本もおすすめ ー";
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「+」
document.write(x);
```

問題3-12 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(); //  $0 \leq x < 1$  になる
x *= 10; // 「x = x*10」と同じ。自分を10倍したものを、自分(x)に記憶。  $0 \leq x < 10$  になる
x = Math.floor(x); // 切り下げる。xは0以上9以下の整数になる
document.write(x);
```

■新しい関数

Math.floor(x) —— xの小数点以下を切り下げた値を返す(x以下の最大の整数を返す)

問題3-13 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

```
// 上の方法で0から9を出し、その後1増やす
let x = Math.random(); //  $0 \leq x < 1$ 
x *= 10; //  $0 \leq x < 10$ 
x = Math.floor(x); // 切り下げる。xは0以上9以下の整数になる
●●● // 【ここで1を追加する操作を行う】 ← ★考えてください★
document.write(x);
```

問題3-14 今日のラッキーナンバー(1~100までのいずれか)を表示するプログラムを作成せよ。1~100までの数字はランダムに出現するものとする

第4章 4.10 練習問題

「for文はややこしいなあ」と感じましたか。そう感じた人も、いくつか問題を解いて自分で使ってみると、「結構便利」と感じられるようになるでしょう。

最初のほうの問題は単純ですので、もし分からなかったら、本文の説明をもう一度読んでみてください。

問題4-9以降はかなり難しいので、「難しすぎる」と思ったら、後ろの章の問題をやってから、再度戻って取り組んでみてください

(for文の問題なので、この章に含めましたが、もう少し後ろの章の問題にしたいような「難易度高め」の問題です)。

問題4-1 「私はプログラミングが好きになる～～」と16行出力するプログラムをforループを使って作成せよ【写経】

```
"use strict";
for (let i=1; i<=16; i++) { // (i=0; i<16; i++) でもOK
  document.write("私はプログラミングが好きになる～～<br>");
}
```

この出力を音読して、自己暗示をかけましょう(^_^)。

問題4-2 問題4-1の出力の、各行の前に、1から順番に番号を振れ【写経】

```
"use strict";
for (let i=1; i<=16; i++) {
  document.write(`${i}. 私はプログラミングが好きになる～～<br>`);
  // 文字列中で変数を使いたい場合は`...` で囲む (テンプレートリテラル)
}
```

```
1. 私はプログラミングが好きになる～～
2. 私はプログラミングが好きになる～～
3. 私はプログラミングが好きになる～～
... 【中略】
15. 私はプログラミングが好きになる～～
16. 私はプログラミングが好きになる～～
```

バリエーション

- 上の問題を1から80まで、80行出力するように変更せよ (**prob0402b.html**)
- 上の問題を1から10万まで、10万行出力するように変更せよ。なお、10万は「**10_0000**」と書くこと (**prob0402c.html**。 「_」については「4.2 同じようなことを繰り返すfor文」参照)

問題4-3 問題4-2で、出力するHTMLコードをすべて変数に保存しておいて、最後に一度だけdocument.writeを呼ぶように変更せよ【写経】

```
"use strict";
let x = ""; // 「空文字列」に「初期化」しておく
for (let i=1; i<=16; i++) {
  x += `${i}. 私はプログラミングが好きになる～～<br>`;
  // できた文字列を x の後ろに次々と追加して行く
}
document.write(x); // 最後に一度だけ書き出す。こうしたほうが少し実行が速い
```

問題4-4 問題4-3で途中で作られるHTMLコードをconsole.logで確認しながら実行するようにしてみよ【写経】

console.logを使った「コンソール」への出力については第8章で詳しく説明しますが、ここでトライしてみてください。

```
11 "use strict";
12 let x = ""; // 「空文字列」に「初期化」しておく
13 for (let i=1; i<=16; i++) {
14   x += `${i}. 私はプログラミングが好きになる~~<br>\n`; // xの最後に追加
15   console.log(`--- ${i}回目のループ ---\n`); // \nは改行のため
16   console.log(x); // xの途中経過を表示
17 }
18 document.write(x); // 最後に一度だけ書き出す。実行が（少し）速くなる
```

上のコードについては次の点に注意してください。

- コード14行目、15行目の「\n」は改行を表す。コンソールに表示するときは、各行の最後で改行したほうが見やすいので、「\n」を出力している（HTMLでいえば
の役割をする）
- 「\」はWindowsでは「¥」と表示されることが多い。macOSのエディタでは「¥」あるいは「option+¥」で入力できることが多い。入力方法がわからない場合は、解答例jsdata/exercise/prob0404.htmlからコピー・ペーストする

VS Codeで実行([実行]→[デバッグなしで実行])すると、ブラウザに結果が表示されると同時に、図4.11のように、VS Code右下の「パネルペイン」(第1章の図1.12参照)の「デバッグコンソール」に、生成されるHTMLコードの途中経過(xの値の変化)が表示されます。

デバッグコンソールの出力を一番最初に戻って追ってみてください。1回ループするごとに、出力が1行ずつ増えていることを確認しましょう。

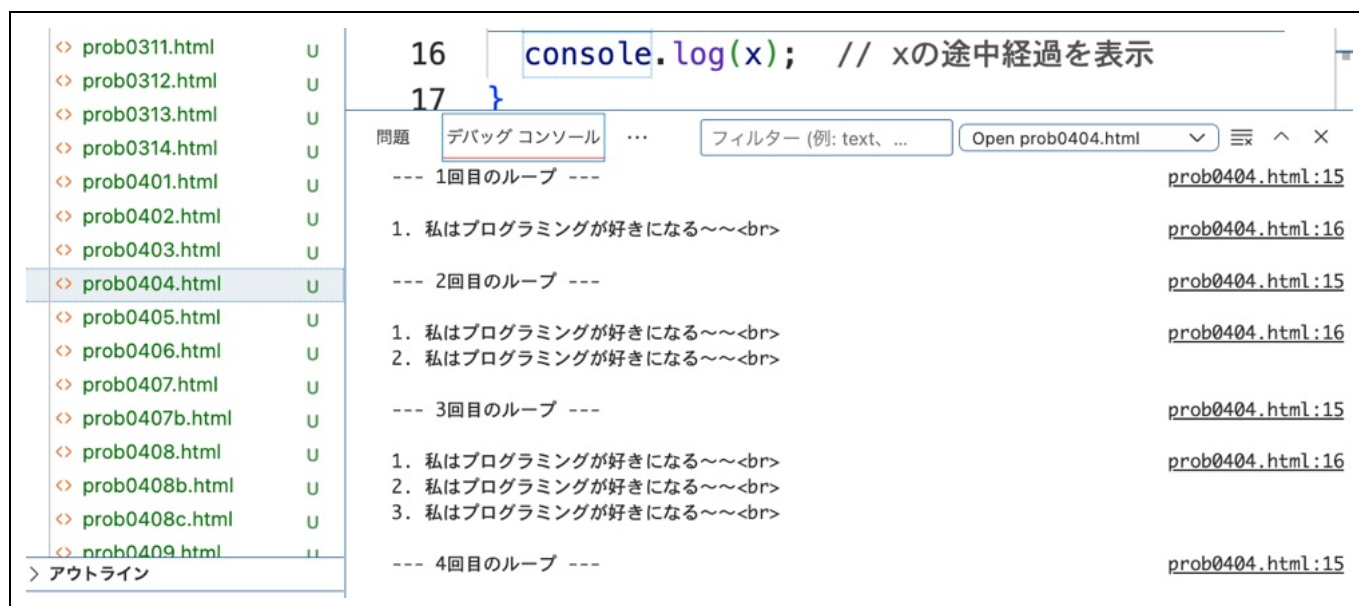


図4.11: VS Codeの「デバッグ コンソール」に途中経過を表示。ループ1回ごとにxの値が増えていく様子がわかる

一方、ブラウザ(Chrome)のドキュメント領域で、右クリック→[検証]→[コンソール]と選択して、ブラウザの「コンソール」を表示してみてください(図4.12)。

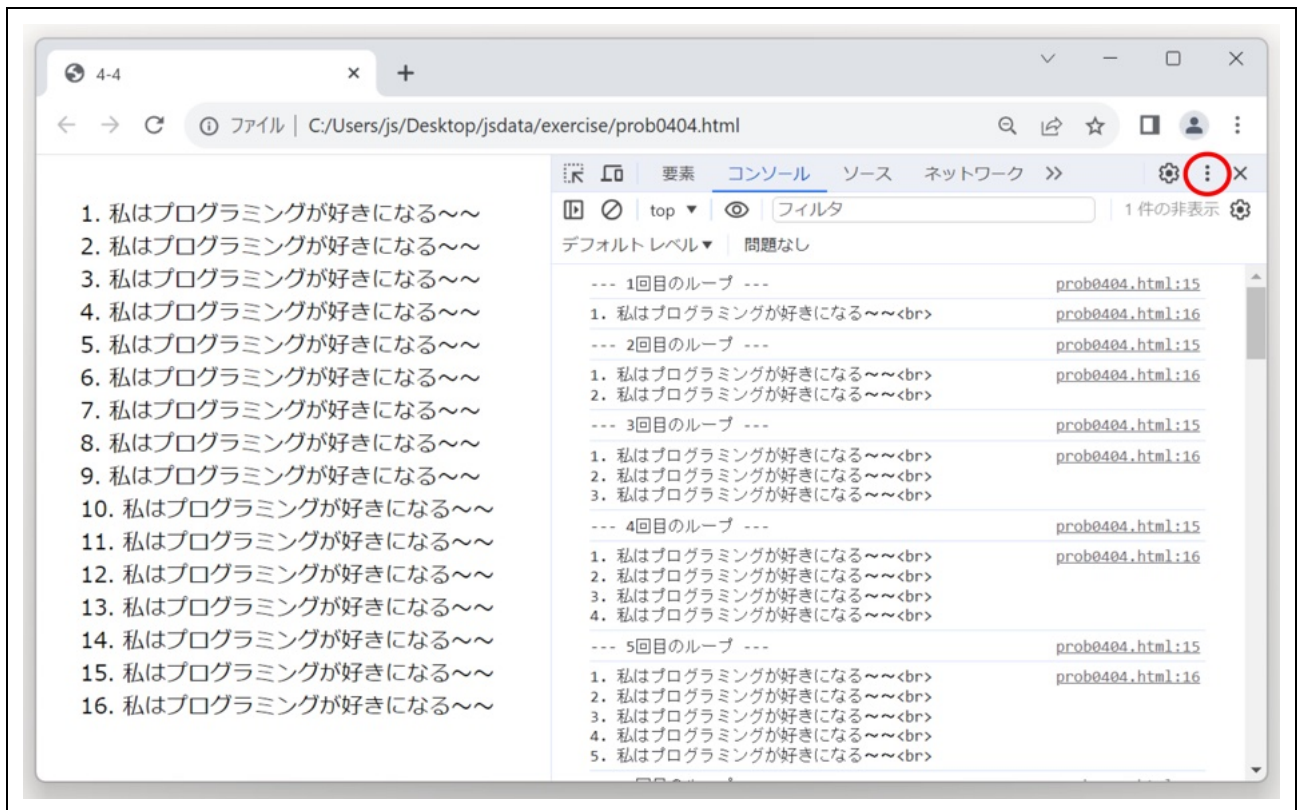


図4.12: Chromeの「コンソール」に途中経過を表示したもの。右上の「ケバブメニュー」(赤丸部分)をクリックすることで、ペインのレイアウトを変更できる

■Note

Chromeのタブの名前が英語(「Console」など)になっているときは「DevTools is now available in Japanese!」というメッセージが表示されていると思います。この場合は[Switch DevTools to Japanese]というボタンをクリックすると、タブ名などが図4.12のように日本語の名前(「コンソール」など)に変わります。

なお、右上の「ケバブメニュー」(図4.12の赤丸部分)をクリックすることで、ドキュメント領域とコンソールのペインのレイアウトを変更できます(別のウィンドウへの表示も可能)。

このChromeの「コンソール」の内容は、VS Codeの「デバッグコンソール」の内容と同じものです。

元来、`console.log`による出力は、ブラウザのコンソールに表示されるはずのものです。VS Codeは、ブラウザのコンソールへの出力を「コピー」して、自分の管理するペインにも出力しているわけです。

利用者にとっては、ブラウザの「コンソール」を開かずに、VS Codeのパネルペインを見るだけでコンソールへの出力内容を確認できるので手間が省けます。VS Codeを使っている人は、今後はChromeのコンソールではなく、VS Codeの「デバッグコンソール」を見るようにしましょう。

■Note

パソコン登場以前の、大型のコンピュータ(「大型機」)などで、コンピュータを制御するためのキーボードとディスプレイを備えた端末のことを「コンソール」と呼んでいました。そこから派生して、Windowsの「コマンドプロンプト」あるいはPowerShellなどや、macOSの「ターミナル」のように、文字による「コマンド」を入力して、いろいろな操作を行うためのウィンドウも「コンソール」と呼ぶことがあります。

ここで使っている、VS Codeの「デバッグコンソール」やChromeの「コンソール」は、途中経過(「ログ」と呼ばれます)の出力のために用いられていますが、第8章で見るように、コマンドを入力してその実行結果を表示させることもできます。

したがって、どちらも「コンソール」の一種です。

問題4-5 問題4-4で、番号を書くのに、数字を直接書くのではなく、番号付きのリストを出力するタグ(...)を使って書け(olタグを使ったほうが数字が右揃えになって見た目がよい)

■HTMLメモ

次のようなHTMLコードを書くことで「順序付きリスト(ordered List)」を出力できます(example/ch0408.html)。liはlist item(リストの要素)の意味です。

```
<ol>
  <li>
    私はプログラミングが好きになる～～
  </li>
  <li>
    私はプログラミングが好きになる～～
  </li>
  ...
  【中略】
  ...
  <li>
    私はプログラミングが好きになる～～
  </li>
</ol>
```

上のHTMLコードをブラウザで表示すると次のように、右揃えで番号がついてリストが表示されます(問題4-2の出力と番号の部分を比較してください)。

```
1. 私はプログラミングが好きになる～～
2. 私はプログラミングが好きになる～～
3. 私はプログラミングが好きになる～～
4. 私はプログラミングが好きになる～～
5. 私はプログラミングが好きになる～～
6. 私はプログラミングが好きになる～～
7. 私はプログラミングが好きになる～～
8. 私はプログラミングが好きになる～～
9. 私はプログラミングが好きになる～～
10. 私はプログラミングが好きになる～～
11. 私はプログラミングが好きになる～～
12. 私はプログラミングが好きになる～～
13. 私はプログラミングが好きになる～～
14. 私はプログラミングが好きになる～～
15. 私はプログラミングが好きになる～～
16. 私はプログラミングが好きになる～～
```

ch0408.htmlこのような「HTMLのコード」を「JavaScriptを使って」書き出せばよいのです。

ヒント

- 繰り返すところ(...)だけをループに入れる
- とは繰り返す必要がない → ループの中に入れてはいけない

問題4-6 picturesフォルダにある画像を8枚連続して表示するプログラムをforループを使って書け【写経】

```
"use strict";
let html = "";
for (let i=0; i<8; i++) {
  const ファイル名 = `pictures/picture00${i}.jpg`;
  const 画像タグ = `
```

問題4-7 問題4-6でスタイルを指定して、画像の大きさを変更してみよ（100px、20%、50%、10cm、...）など。なお、解像度の高いモニタで"cm"、"mm"などで指定すると、画面上の大きさは指定した大きさにならないことがある（プリンタで出力すれば大丈夫。詳しくは次のページなどを参照 <https://musha.com/scjs?ln=0401>）

```
"use strict";
const スタイル = `style="width: ●●;"`; // ←★要変更★

let html = "";
for (let i=0; i<8; i++) {
  const ファイル名 = `pictures/picture00${i}.jpg`;
  const 画像タグ = `
```

問題4-8 forループとHTMLの<table>...</table>を使って1行4列の画像のテーブルを作成せよ。画像の大きさは120pxとする【写経】

まずHTMLのtableタグについて説明しましょう。

■HTMLメモ

次のようなHTMLコードを書くことで表(table)を作ることができます(example/ch0409.html。このほかにも表関連のタグがありますが、ここでは省略します。ここで用いられているタグだけで表を作ることができます)。

```
<table>
  <tr>  <!-- 1行目の始まり。 trは Table Row の略 -->
    <td>1-1</td>  <!-- 1行目の1列目の要素。 td は Table Dataの略 -->
    <td>1-2</td>  <!-- 1行目の2列目の要素 -->
    <td>1-3</td>
    <td>1-4</td>
  </tr>  <!-- 1行目の終わり -->
  <tr>  <!-- 2行目 -->
    <td>2-1</td>  <!-- 2行目の1列目の要素 -->
    <td>2-2</td>
    <td>2-3</td>
    <td>2-4</td>
  </tr>  <!-- 2行目の終わり -->
  <tr>  <!-- 3行目 -->
    <td>3-1</td>
    <td>3-2</td>
```

```

        <td>3-3</td>
        <td>3-4</td>
    </tr>    <!-- 3行目の終わり -->
</table>

```

上のコードを表示すると次のように表示されます。

1-1、1-2、...に画像(**img**)タグを書けば、画像が表のように並ぶことになります。

1-1	1-2	1-3	1-4
2-1	2-2	2-3	2-4
3-1	3-2	3-3	3-4

この問題では画像を1行4列に並べるので、次のようなHTMLコードを出せばよいことになります。

```

<table>
  <tr>
    <td></td>
    <td></td>
    <td></td>
    <td></td>
  </tr>
</table>

```

上のコードを出力するJavaScriptのプログラム【写経用】

- ループ変数（forループの制御部分で使う変数）は列番号なので、画像番号を表す変数を別に用意して、ループするたびに更新する必要があることに注意

```

"use strict";
const スタイル = 'style="width: 120px;";';
let 画像番号 = 0;
let html = "<table>";
html += "<tr>";
for (let 列番号=0; 列番号<4; 列番号++) { // 列
  const ファイル名 = `pictures/picture00${画像番号}.jpg`;
  画像番号++;
  html += `<td>  </td>`;
}
html += "</tr>";
html += "</table>";
document.write(html);

```

第4章のこれ以降の問題は、少し難しいので、最初はスキップしても結構です。

問題4-9 forループとHTMLの<table>...</table>を使って2行4列の画像のテーブルを作成せよ



出したいHTMLコード(一例。スタイル指定は省略)

```
<table>
<tr>
  <td></td>
  <td></td>
  <td></td>
  <td></td>
</tr>
<tr>
  <td></td>
  <td></td>
  <td></td>
  <td></td>
</tr>
</table>
```

上のコードを出力するJavaScriptのプログラム(その1。一重のループを2回)

```
"use strict";
const スタイル = `style="width: 200px;"`;
let 画像番号 = 0; // 出力する画像の番号を記憶する

let html = "<table>";

html += "<tr>"; // 1行目
for (let 列番号=0; 列番号<4; 列番号++) {
  let ファイル名 = `pictures/picture00${画像番号}.jpg`;
  画像番号++;
  html += `<td>  </td>`;
}
html += "</tr>";

html += "<tr>"; // 2行目
for (let 列番号=0; 列番号<4; 列番号++) { // ←**要変更**
  let ファイル名 = `pictures/picture00${画像番号}.jpg`;
  画像番号++;
  html += `<td>  </td>`;
}
html += "</tr>";

html += "</table>";
```

```
document.write(html);
```

上のコードを出力するJavaScriptのプログラム(その2:二重のループを使う)。こちらのほうが高度

```
"use strict";
const スタイル = `style="width: 120px;"`;
let html = "<table>";
let 画像番号 = 0; // 出力する画像の番号を記憶する
for (let 行番号=0; 行番号<2; 行番号++) { // 行 (外側のループ)
  html += "<tr>";
  for (let 列番号=0; 列番号<4; 列番号++) { // 列 (内側のループ)
    html += "<td>";
    const ファイル名 = `pictures/picture00${画像番号}.jpg`;
    画像番号++; // 次の画像の番号を準備する
    console.log(ファイル名);
    html += ``;
    html += "</td>";
  }
  html += "</tr>";
  // console.log(html) // ←ここで出力すると1回ループを回った時の結果がわかる
}
html += "</table>";
document.write(html);
```

ループをするたびに変数の値は次のように変化する

行番号	列番号	画像番号
0	0	0
0	1	1
0	2	2
0	3	3
1	0	4
1	1	5
1	2	6
1	3	7

- 最初のうち、プログラムを読むときには上のような表を作って変数の値の変化を確認していくとよい
- 二重のループを使うことで、同じようなコードを繰り返し書かなくて済む
- 変更にも強くなる —— 似たようなコードが複数あると、「直し間違い」や「直し忘れ」が生じがち

問題4-10 forループとHTMLの<table>...</table>を使って4行4列の画像のテーブルを作成せよ



```
"use strict";
const 行数 = 4; // こうしておけば、ここを変えるだけで行数の変化に対応可能
let html = "<table>";
let 画像番号 = 0; // 出力する画像の番号を記憶する
for (let 行番号=0; 行番号<行数; 行番号++) { // 行 (外側のループ)
  html += "<tr>";
  for (let 列番号=0; 列番号<4; 列番号++) { // 列 (内側のループ)
    html += "<td>";
    let ファイル名 = `pictures/picture00${画像番号}.jpg`;
    if (10 <= 画像番号) {
      ファイル名 = ●●●●; // ←★★要変更★★
    }
    画像番号++; // 次の画像の番号を準備する
    html += ``;
    html += "</td>";
  }
  html += "</tr>";
}
html += "</table>";
document.write(html);
```

問題4-11 forループとHTMLの<table>...</table>を使って4行4列の画像のテーブルを作成し、クリックしたら大きな画像を表示するページに移動するようにせよ

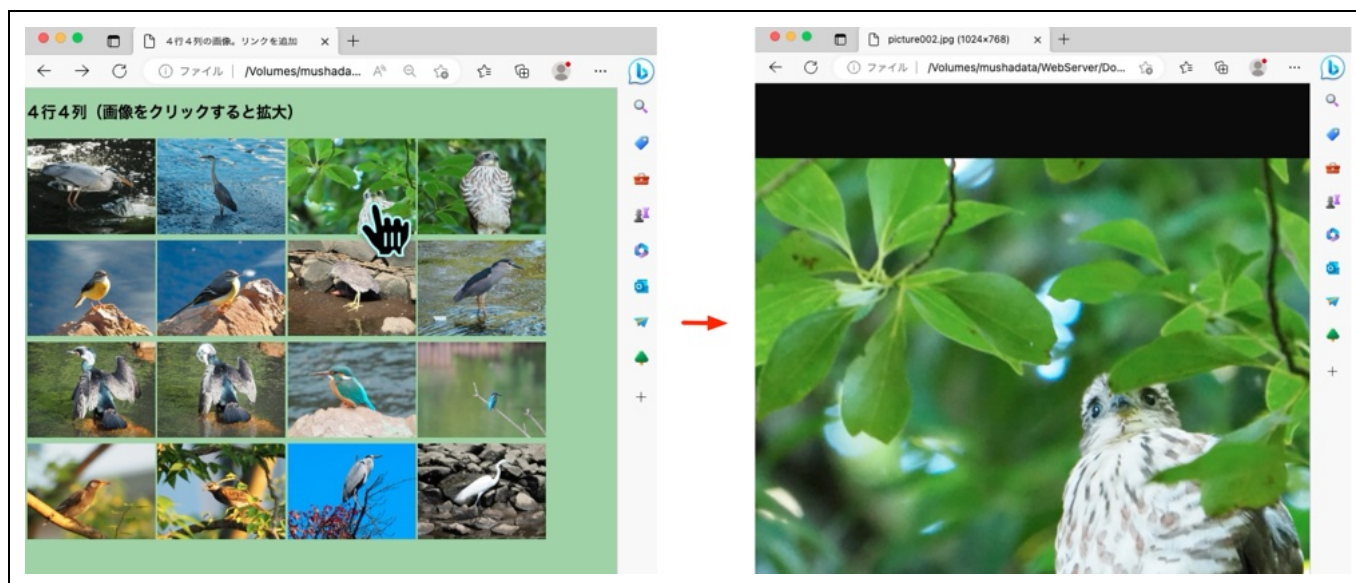


図4.13: 画像をクリックするとその画像のページに移動する

■HTMLメモ

HTMLの「リンク」を書くにはaタグを使います^{*5}。形式は次のとおりです。

```
<a href="【URL】">【リンク対象】</a>
```

- **【URL】** —— 外部ページのアドレス (**https://...**) を指定することも、自分のサイトのほかのファイル (ページ) を指定することもできます
- **【リンク対象】** —— この部分がリンクになり、選択 (クリックあるいはタップ) すると**【URL】**に移動します。テキストだけでなく画像なども指定できます

この問題では、画像クリックすると、その画像のファイルに移動すれば、その画像だけがブラウザに大きく表示されます。「**pictures**フォルダの**picture002.jpg**というファイル」を表の1項目として表示して、クリックしたらその画像のページに移動したければ、次のコードを書きます。

```
<a href="pictures/picture002.jpg"></a>
```

これで画像をクリックすると、その画像だけがドキュメント領域に大きく表示されます(図4.13)。

^{いかり}
[*5] aタグのaは、「錨」「錨で固定する」などの意味をもつanchorの先頭文字です。このためaタグは「アンカータグ」とも呼ばれます。「リンク先のページにつなぎとめる」といった意味合いからこの単語が選ばれたようです。リンク先を示す**href**は「hypertext reference」の省略形です。hypertextはウェブページのようなリンクが張られた文章(テキスト)を意味する言葉で、referenceは「参照」などの意味をもつ単語です。

問題4-12 下に説明する`toString`と`padStart`を使って`picture000.jpg`から`picture039.jpg`までの40枚の画像を表示して、それぞれの画像を選択した際にオリジナルの画像を大きく表示するようにせよ。また、あとで枚数が増えたときに簡単に変更できるよう、冒頭で定数「画像の枚数」を使って画像の枚数を指定するようにせよ

`toString`と`padStart`

問題4-10などのコードでは、画像のファイル名を作るのにif文を使って「0から9までのとき」と「10以上のとき」を分けていますが、関数`toString`と`padStart`を使うとif文を使わずにこの処理ができます。

たとえば、次のようなコードを書けば`pictures`フォルダにある`picture000.jpg`から`picture015.jpg`までの画像を表示できます(`example/ch0410.html`)。

```
"use strict";
const スタイル = `style="width: 25%;`
let html = "";
for (let i=0; i<=15; i++) {
  const 数字部分 = i.toString().padStart(3, "0");
  // toStringは整数 (i) を文字列にしてくれる
  // その結果を受けてpadStartは、先頭 (Start) に0を埋めて (padして)、3桁にする
  // その結果、数字部分は001, 002, 003, ..., 009, 010, 011, ...となる
  const ファイル名 = `pictures/picture${数字部分}.jpg`;
  const 画像タグ = `
```

`example/ch0410.html`のコードを参考に、問題4-12のコードを作成してください。

このように、知識があれば問題をより簡単に解決できる場合もあります。

なお、`padStart`は先頭部分を指定の文字(上の場合「0」)で埋めますが、逆に最後の部分を指定の文字で埋めてくれる`padEnd`もあります。

■新しい関数

`toString()` —— 数字を文字列に変換する。文字列に変換することで、文字列に用意されている関数(メソッド)を使えるようになる

■新しい関数

`padStart(n, c)` —— 先頭部分を第2引数に指定した文字`c`で埋めて、第1引数`n`に指定した文字数の文字列を作る

次章の練習問題には、ユーザー定義関数とループを組み合わせる問題があります。

問題5-1 引数xに対してxの2乗を返す関数「自乗を計算」を作り、これを呼び出して10, 4, 9, 12, 128の2乗を計算せよ【写経】

```
function 自乗を計算(x) {  
    return x * x;  
}  
  
document.write(自乗を計算(10) + "<br>");  
document.write(自乗を計算(4) + "<br>");  
document.write(自乗を計算(9) + "<br>");  
...
```

問題5-2 2つの引数を与えられて、その平均を返す関数「平均を求める」を作り、これを呼び出して次の組の平均を求めよ —— 10と6、4と4、130と80、5.2と4.6

```
function 平均を求める(a, b) {  
    return ●●  
}  
  
document.write(平均を求める(10, 6) + "<br>");  
document.write(平均を求める(4, 4) + "<br>");  
document.write(平均を求める(130, 80) + "<br>");  
document.write(平均を求める(5.2, 4.6) + "<br>");
```

問題5-3 引数にファイル名を指定すると、その画像を640px × 480pxの大きさで表示するタグを作って返してくれる関数「画像タグを生成」（あるいは「makeImageTag」）を作ってもうまく動作するかテストせよ【写経】

次の問題で利用するので、タグが正しく作られて返ってくることをconsole.logやalertなどで確認する。**画面に画像は表示されなくてOK。**

```
// `pictures/picture000.jpg` →  
// `  
let x = 画像タグを生成("pictures/picture000.jpg");  
console.log(x);  
  
function 画像タグを生成(ファイル名) {  
    let html = `    return html; // できあがった<img>タグを呼び出し側に返す  
}
```

補足説明

1. 関数「画像タグを生成」の先頭（引数の部分）で次の文に相当する代入が行われる

ファイル名 = "pictures/picture000.jpg"

2. それから関数の残りの部分が実行される（変数「ファイル名」には**"pictures/picture000.jpg"**が入っている）

```
let html = ``;
return html;
```

3. 関数側の変数**html**の値が、関数を呼び出した側の変数**x**に代入される

問題5-4 picturesフォルダが例題のファイルと同じフォルダにあることを確認してから、前の例題のconsole.logの下に、document.writeを加えて、できあがったタグを書き出し、プログラムを実行して、画像が表示されることを確認せよ【写経】

```
let x = 画像タグを生成("pictures/picture000.jpg");
console.log(x);
document.write(x);

function 画像タグを生成(ファイル名) {
  let html = ``;
  return html; // htmlの値を呼び出した側に返す
}
```

バリエーション

- 上で作成した関数「**画像タグを生成**」で**width**や**height**の指定をいろいろ変えて、画像の大きさが変わることを確認せよ。たとえば「高さ300px」を指定してみよ（高さの指定は「**height: ●●px;**」）
- 「**画像タグを生成**」を呼び出すときの引数のファイル名を変えて、画像が変わることを確認せよ（たとえば **pictures/pictures002.jpg**）

問題5-5 問題5-3で定義した関数「画像タグを生成」を変更して、画像の幅を指定する引数を渡せるようにした「幅を指定して画像タグを生成」を作成せよ

```
let x = 幅を指定して画像タグを生成("pictures/picture000.jpg", 320);
document.write(x);

function 幅を指定して画像タグを生成(ファイル名, 幅) {
  // ファイル名 -- 画像のファイル名 (第1引数)
  // 幅 -- 画像の幅をピクセル (px) 単位で指定 (第2引数)
  let html = ``; // ←★要変更★
  return html;
}
```

バリエーション

- 指定する幅を色々変えて、画像の大きさが変わることを確認せよ

問題5-6 問題5-5の関数「幅を指定して画像タグを生成」を使って、画像の幅を100ピクセル、200ピクセル、300ピクセル、...、800ピクセルに指定した画像を連続して表示せよ。同じ行には1つずつ表示すること



ヒント

- forループを使う

問題5-7 上と同じように大きさを変えて8枚の画像を表示するが、画像のファイルもpicutre000.jpg、picture001.jpg、picture002.jpg、...、picture007.jpgと順に変化させよ。同じ行には1つずつ表示すること（下図参照）



問題5-8 関数「幅を単位付きで指定して画像タグを生成」を定義して、画像の幅を「ドキュメント領域の横幅」の10%、20%、30%、40%に指定して画像を連続して表示せよ。画像もpicutre000.jpg、picture001.jpg、picture002.jpg、...、picture007.jpgと変化させよ。同じ行には4つずつ表示すること（下図参照）



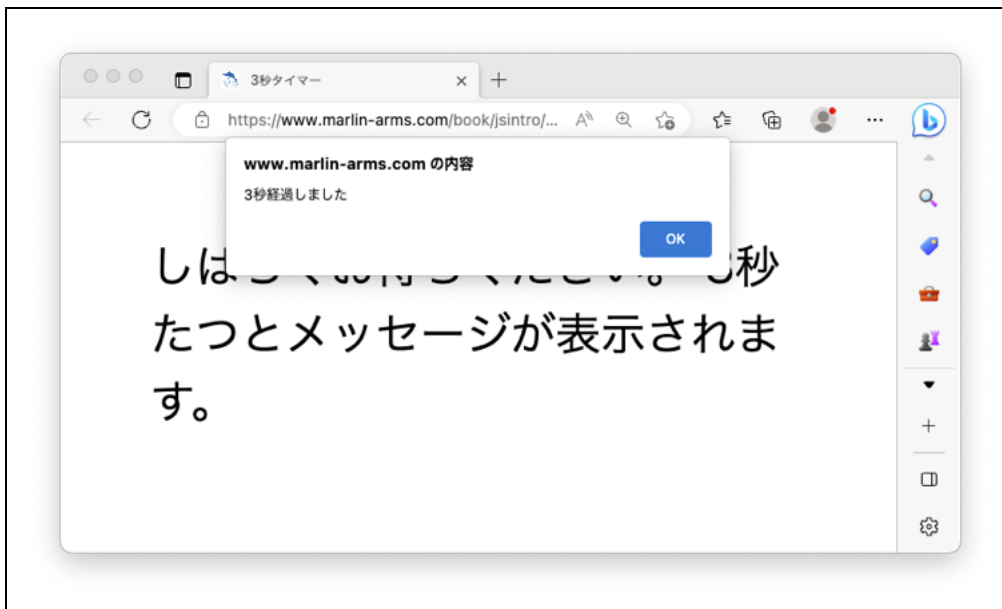
第6章 6.9 練習問題

関数`setInterval`と第8章以降で説明する関数を組み合わせることで簡単な「アニメーション」が作れます。アニメーションの問題は第11章にありますので、あとで挑戦してみてください。

無名関数やアロー関数は以降の章で繰り返し登場しますので、この章では基本的な問題だけを解いてみましょう。

問題6-1 ページを読み込んでから3秒後に、「3秒経過しました」というダイアログボックスを一度だけ表示せよ。関数`setTimeout`を使うこと

`setInterval`は繰り返しの実行に利用しますが、何秒か後に一度だけ実行したいというときは、`setTimeout`を用います。`setTimeout`は第2引数に指定された時間（ミリ秒単位）経過後に、1度だけ第1引数で定義された関数を実行します（詳しい説明は教科書第6章p. 127を参照してください）。



```
16 <script>
17   "use strict";
18   setTimeout(メッセージを表示, 3*1000);
19   // 第2引数は「3000」と書いてもOKだが、「3*1000」のほうが3秒であることが明確
20   // ほんの少しだけ実行時間は多くなると思われるが、人間にとっては無視できる長さ
21   // なので、わかりやすさを優先
22
23   function メッセージを表示() {
24     ●●; // ←*要変更*
25   }
26 </script>
```

問題6-2 問題6-1の「ダイアログボックスを表示」を無名関数に変更せよ

その場で関数を作りたい時、名前はいらない時は「無名関数」を使うと便利。上の「メッセージを表示」を「展開」してその場を書く。

問題6-3 問題6-2の無名関数をアロー関数を使って書け

問題6-4 ページを読み込んでから3秒後に、「3秒経過しました」というダイアログボックスを表示し、以降3秒経過するたびに同じダイアログボックスを表示せよ。ただし3回でやめるものとする

```
15 <script>
16 "use strict";
17 const 秒数 = 3;
18 const 繰り返す回数 = 3;
19 let カウンタ = 0;
20 const タイマーID = setInterval(メッセージを表示, 秒数 * 1000);
21
22 function メッセージを表示() {
23   カウンタ++;
24   alert(秒数 + "秒経過しました");
25   if (繰り返す回数 <= カウンタ) {
26     // 「===」を使ってもよいが、「<=」のほうが安全
27     // プログラムを変えたときに、===が成り立たなくなる危険を回避できる
28     // <= を使っておけば、===が真にならなくても、大きくなりさえすれば停止する
```

```
29     clearInterval(タイマーID);
30   }
31 }
32 </script>
```

バリエーション

- 無名関数やアロー関数を使って書いてみよ

第7章 7.5 練習問題

解答例は、第1章でダウンロードしたフォルダの `jsdata/exercise` に入っています。

問題7-1 配列の内容を順番に書き出すプログラムを作れ【写経】

```
const 色の配列 = ["赤", "青", "黄色", "緑", "紫", "黒", "ピンク", "茶色", "灰色"];
for (let i=0; i< 色の配列.length; i++) { // 「配列名.length」で配列の大きさ
  document.write(色の配列[i] + "<br>");
}
```

【別解】

for文には次のようなキーワード `of` を伴う形式もあります。変数「色」には「色の配列」の要素が順番に入ります。この構文では、配列の添字(上の `i`)のことを考える必要がなくなるので、全部の要素について何かを行う場合に便利です (`prob0701b.html`)。

```
const 色の配列 = ["赤", "青", "黄色", "緑", "紫", "黒", "ピンク", "茶色", "灰色"];
for (const 色 of 色の配列) {
  document.write(色 + "<br>");
}
```

問題7-2 今日のラッキーカラー（赤、青、黄色、緑、紫、黒、ピンク、茶色、灰色のいずれか）をランダムに表示するプログラムを配列を使って書け【写経】

あなたの今日のラッキーカラーは紫です

- 配列名.length で配列の大きさがわかる
- 0以上n以下の整数をランダムに得るには下に書いた「ランダムな整数を生成」を利用する（第3章の問題3-12も参考に。少し難しいので、コードの意味がよく理解できなかったら「棚上げ」を。とにかく、関数「ランダムな整数を生成」を呼べば0以上n以下の整数がランダムに返ってくる）


```
const 色の配列 = ["赤", "青", "黄色", "緑", "紫", "黒", "ピンク", "茶色", "灰色"];
const i = ランダムな整数を生成(色の配列.length-1); // lengthで配列の大きさ
document.write(`あなたの今日のラッキーカラーは${色の配列[i]}です`);

function ランダムな整数を生成(n) { // 0以上n以下の整数をランダムに得る
  let x = Math.random(); // たとえば n が 10 ならば
  x = x * (n+1); // 11倍することになるので 0 ≤ x < 11 になる
  x = Math.floor(x); // 小数点以下を切り下げるので 0 ≤ x ≤ 10 の整数になる
  return x;
}
```

問題7-3 今日のラッキーカラー（赤、青、黄色、緑、紫、黒、ピンク、茶色、灰色のいずれか）をランダムに表示するプログラムを書け。ラッキーカラーの文字はその色で表示すること

```
const 色の配列 = ["赤", "青", "黄色", "緑", "紫", "黒", "ピンク", "茶色", "灰色"];
const 色指定文字列の配列 = ["red", "blue", "yellow", "green", "purple", "black",
  "pink", "brown", "gray"];
const i = ランダムな整数を生成(色の配列.length-1); // length は配列の大きさ
const 色 = 色の配列[i];
const 色指定文字列 = 色指定文字列の配列[●●]; // ●●の部分は考えて変える
let メッセージ =
  `あなたの今日のラッキーカラーは<span style="color: ${色指定文字列}">${色}</span>です。`;
document.write(メッセージ);

// 文字の色の指定はたとえば次のように指定する — <span style="color: red;">赤</span>
```

問題7-4 今日のラッキーカラー（赤、青、黄色、緑、紫、黒、ピンク、茶色、灰色のいずれか）をランダムに表示するプログラムを書け。ラッキーカラーの文字はその色で表示すること。ただし、黄色は見にくいので適当な背景色（background-color）を指定すること

問題7-5 今日のラッキーアイテム（マウス、万年筆、シャープペン、スマートフォン、ホッチキス、ポストイット、定期券、ペンダントのいずれか）を表示するプログラムを作成せよ

今日のあなたのラッキーアイテムは **定期券** です。

問題7-6 下のようなメッセージを表示するプログラムを作成せよ*4【写経】

```
タイモンが1匹来ました。合計1匹になりました。
タイモンが2匹来ました。合計3匹になりました。
タイモンが3匹来ました。合計6匹になりました。
タイモンが4匹来ました。合計10匹になりました。
..... <以下同様>
タイモンが10匹来ました。合計∞匹になりました。
```

タイニー モンスター

【*4】「タイモン」は tiny monsterの省略形で、この本の中に棲む仮想の動物です(^o^).

ヒント

覚えておく必要があるもの

1. 新たに何匹来たか (1, 2, 3, ...)
2. 合計何匹いるか (これまでの「合計」に新しく来た数を加える)

合計 += i;

コード例

```
"use strict";
let 合計 = 0;
for (let i=1; i<=10; i++) { // この問題の添字は (0からではなく) 1から始めたほうが楽でしょう
  合計 += i;
  document.write(`タイモンが${i}匹来ました。`);
  document.write(`合計${合計}匹になりました。<br>`);
}
```

問題7-7 下のようなメッセージを表示するプログラムを作成せよ (1匹の時だけ特別処理)

```
タイモンが1匹来ました。
タイモンがもう2匹来ました。合計3匹になりました。
タイモンがもう3匹来ました。合計6匹になりました。
タイモンがもう4匹来ました。合計10匹になりました。
タイモンがもう5匹来ました。合計15匹になりました。
.....
.....
タイモンがもう100匹来ました。合計●●匹になりました。
```

問題7-8 [ダイアログボックスを使った入力] ユーザーが入力した整数をそのまま出力するプログラムを作成せよ【写経】

例1:ユーザーが「10」を入力 → 「10」を入力しましたね

例2:ユーザーが「100」を入力 → 「100」を入力しましたね

```
"use strict";
const x = prompt("整数を入力してください", 10); // ユーザー入力を得る。10はデフォルト
document.write(`「${x}」を入力しましたね。`);
```

■新しい関数

prompt(message, default) —— 第1引数にしてしたmessageをダイアログボックスに表示し、ユーザーが入力した文字列を戻り値として受け取る。ユーザーが入力する文字列のデフォルト(そのままEnter[return]キーを押したときに使われる値)としてdefaultが表示される

問題7-9 ユーザーに1以上の整数を入力してもらい、1からその整数まで順番に足した値を表示するプログラムを作成せよ

例1:ユーザーが「10」を入力 → 1から10までの合計は55です。

例2:ユーザーが「100」を入力 → 1から100までの合計は5050です。

手順

1. ユーザーからの入力を得る — `x = prompt("整数を入力してください", 10);`
2. 合計を出す — forループを使う

```
"use strict";
x = prompt("整数を入力してください", 10);
let 合計 = 0; // 合計を記憶
for (let i=1; i<=x; i++) {
  合計 += i;
}
document.write(`1から${x}までの合計は●●です`); // ←**要変更**
```

【別解】

じつは、forループを使わずに簡単に求められます。1からnまでの合計は、 $n * (n + 1) / 2$ で求められます。(中学校で習う)「1からnまでの和を求める公式」を使えばよいのです。忘れた人はウェブを検索!(たとえば<https://mathwords.net/1karannowa>)。

```
"use strict";
let n = prompt("整数を入力してください", 10);
if (0 < n) {
  const x = `1から${n}までの合計は${(1+n)*n/2}です`;
  document.write(x);
}
else {
  document.write("計算できません");
}
```

このように、より良い(賢い)方法を知っていれば、簡単に解が求められる場合もあるので、**基本的なアルゴリズム**(処理の手順。詳しくは第13章の「フロー制御とアルゴリズム」参照)を**学んでおくことも有用です**。

■Note

全角の数字が入力されたり、数字以外の文字が入力されたりすると、このプログラムではうまく動きません。本当は、そういった場合でも対応できるようにする必要があります。ここでは、ひとまず、半角の整数が入力されると仮定しておきます。なお、付録Cの練習問題B-2で、全角の数字にも対応したバージョンを作成します。

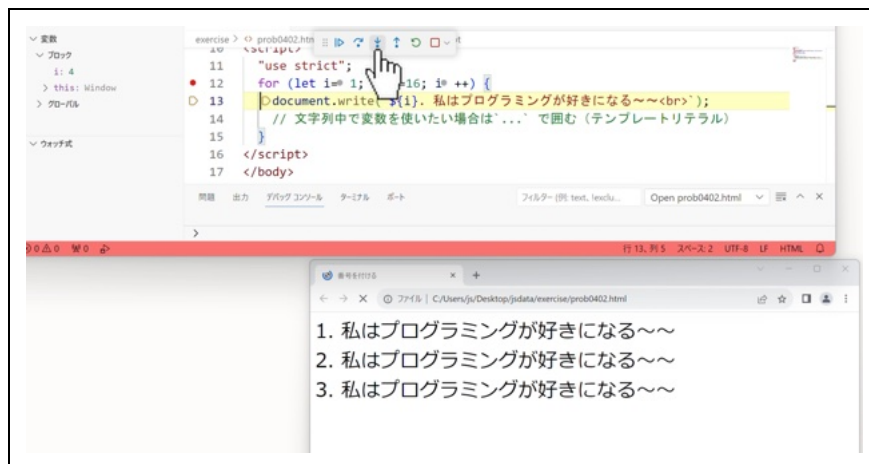
第8章 8.15 練習問題

この章の練習問題は、2問だけですが、次章以降で問題を解くときに、この章で学んだことを活かしてデバッグをしてみてください。

問題8-1 第4章の練習問題4-2の解答例 (`exercise/prob0402.html`) をデバッガを使って1ステップずつ実行し、for文の実行の様子を確認せよ

1. `exercise/prob0402.html`の12行目のfor文の先頭行にブレークポイントを設定する

2. [実行] → [デバッグの実行]（ショートカット：F5）で実行を開始する
3. 12行目で停止したら、必要ならばブラウザのウィンドウをVS Codeで隠れない位置に移動する
4. ステップインのボタンを押しながら、次を確認する
 - * 実行ポイント（カーソルの位置）の移動の様子
 - * VS Codeの左側のサイドバーに表示される変数*i*の値
 - * ブラウザの表示内容



このように、デバッガを使うことで、第4章で説明したfor文の実行の様子を自分の目で確認できます。

同じように、ほかのプログラムについても自分の想定どおり動くか、確認しながら実行できます。

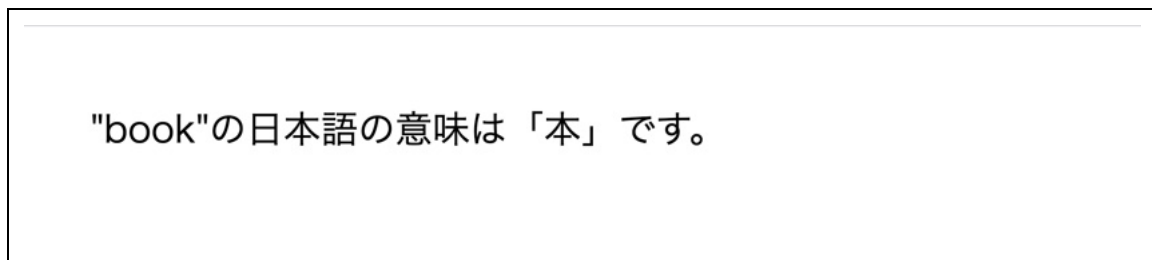
問題8-2 第4章の練習問題4-4の解答例（exercise/prob0404.html）をデバッガを使って1ステップずつ実行し、問題8-1と同様、for文の実行の様子を確認せよ

第9章 9.6 練習問題

この章の練習問題では、「オブジェクト」の基本的な性質を理解するための問題を解いていただきます。動画関連の問題は、イベント処理と関連させたほうが面白いので、第12章にまとめました。しばらくお待ちください。

解答例は、第1章でダウンロードしたフォルダの `jsdata/exercise` に入っています。

問題9-1 ダイアログボックスに英単語を入力すると、その日本語訳をドキュメント領域に表示する、「辞書」ページを作れ。単語数は5語以上とする。辞書にない単語を入力したら「わかりません」と表示する【写経】



```
"use strict";
const 英和辞書 = {
  library: "図書館", mountain: "山", river: "川",
  bug: "虫", warrior: "武者", knight: "騎士",
  star: "星", moon: "月", ant: "蟻",
  elephant: "象", bird: "鳥", book: "本"
};

/* **コメントは入力不要です.**
   「英和辞書」オブジェクトの「library」プロパティの値が "図書館" になる。

   次のように書くこともできる（ただし、上のほうが簡単）
   let 英和辞書 = []; // 「let 英和辞書 = new Array();」でもOK
   英和辞書['library'] = "図書館";
   英和辞書['mountain'] = "山";
   ...
   英和辞書['book'] = "本";
*/

let 英単語 = prompt("英単語を入力してください", "");
// console.log(英単語); // デバッグ用
let 日本語 = 英和辞書[英単語];

let message;
if (! 英単語) { // 何も入力されないと 英単語 の値は""となり「偽」と判定される
  // 「!」がつくと反対になるので、ifの条件は「真」となる
```

```
// 何も入力されなかった場合
message = "英単語を入力してください。"
}
else if (日本語) {
    message = `"${英単語}" の日本語の意味は「${日本語}」です。`;
}
else {
    message = `"${英単語}" の日本語の意味はわかりません。すみません。`;
}
document.write(message);
```

■Note

英和辞書["book"]のように、配列の添字(インデックス)に文字列を指定するもののことを、「連想配列」「辞書」「ハッシュテーブル」などと呼び、多くの言語で使えるようになっています。JavaScriptの場合は、オブジェクトのプロパティを使うことで連想配列が実現できます。

オブジェクトのプロパティなので、「英和辞書["book"]」を「英和辞書.book」と書いても同じことになります。

問題9-2 問題9-1で作った英和辞書の項目の「英単語と訳語」のペアを、ドキュメント領域とコンソールにすべて書き出すプログラムを作成せよ【写経】



コード例

```
"use strict";
const 英和辞書 = {
  library: "図書館", mountain: "山", river: "川",
  bug: "虫", warrior: "武者", knight: "騎士",
  star: "星", moon: "月", ant: "蟻",
  elephant: "象", bird: "鳥", book: "本"
};

for (const 英単語 in 英和辞書) { // 英和辞書のすべての「名前」について実行する
  document.write(`${英単語}: ${英和辞書[英単語]}<br>`); // 改行するのに<br>が必要
  console.log(`${英単語}: ${英和辞書[英単語]}`); // こちらは自動的に改行する
}
```

■Note

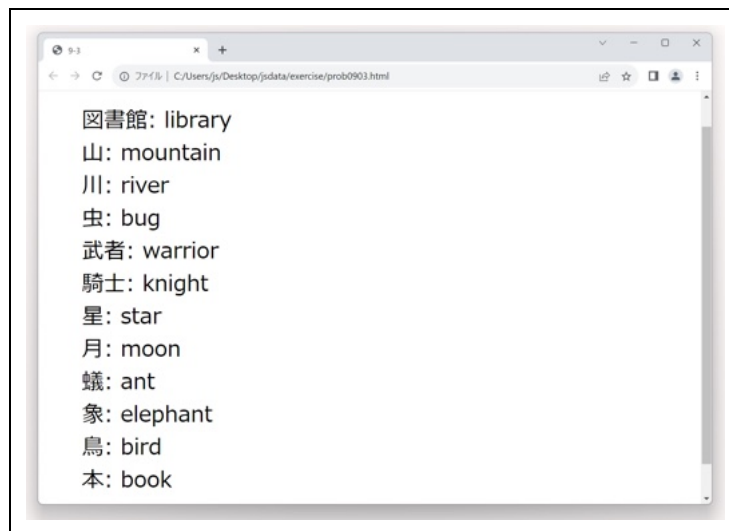
上のコードの「**for (const 英単語 in 英和辞書) {...}**」のコードは別の種類の**for**文です。

オブジェクト「英和辞書」のすべてのプロパティについて、順番にfor文の本体部分{...}を実行することになります。その結果、library、mountain、...、bookに関して、自分自身（プロパティ名、英語）とその値（日本語訳）がすべて表示されます。

なお、表示される順番は不定です（決まっています）。重要なのは対応関係であって、普通の配列と違って何番目に表示されるかは本質的ではありません。

第7章の問題7-1で見た、**for ... of**の構文は、配列の各要素についてループするものですが、**for ... in**の構文は対象がオブジェクトです。

問題9-3 問題9-2で作った英和辞書の項目の「英単語と訳語」のペアを逆にして、和英辞書を作れ



```
"use strict";
const 英和辞書 = {
  library: "図書館", mountain: "山", river: "川",
  bug: "虫", warrior: "武者", knight: "騎士",
  star: "星", moon: "月", ant: "蟻",
  elephant: "象", bird: "鳥", book: "本"
};
```



```
const 和英辞書 = {};
for (const 英単語 in 英和辞書) {
  const 日本語 = 英和辞書[英単語];
  和英辞書[日本語] = ●●; //← ●●を変更する！
  document.write(`${日本語}: ${和英辞書[日本語]}<br>`);
}
```

問題9-4 次のようにあいさつを表示せよ

- 午前5時以降11時より前の時は「おはようございます」
- 午前11時以降、午後6時より前の時は「こんにちは」
- それ以外の時は「こんばんは」

ヒント

- Dateオブジェクトを使う（システムにあらかじめ組み込まれている）。使い方は次のコード参照
- パソコンのタイムゾーン（時間帯）を変更すると現在時刻が変わるので、今の時刻だけでなく他の時刻でもうまくいくか試せる

```
const now = new Date(); // 日時を表すオブジェクト。システムが用意してくれている
const hour = now.getHours(); // 「時」（24時間制）を返す
if (5 <= hour && hour < 11) { // 5:00から11:00前まで
  document.write("...");
}
else if (11 <= hour && hour < 18) { // 11:00から18:00の前まで
  ...
}
...
```

問題9-5 現在の年月日、曜日、時刻（秒まで）を表示せよ

現在 2023年11月10日（金曜日） 21時7分13秒 です。

ヒント

- Dateオブジェクトのメソッドを使う

```
// それぞれ数字を返す
// **月は、0で1月、1で2月、...、11で12月を表すので注意！**

const now = new Date(); // nowは現在時刻を表す
const year = now.getFullYear(); // 年
const month = now.getMonth(); // 0: 1月 1: 2月 ... ★注意！★
const date = now.getDate(); // 日
const dayOfTheWeek = now.getDay(); // 0:日曜 1:月曜 2:火曜 ...
const hour = now.getHours(); // 時（24時間制）
const minute = now.getMinutes(); // 分
const second = now.getSeconds(); // 秒
```

問題10-1 この章の課題の「スライドショー」で画像を1度に2枚ずつ表示するようにせよ。更新する際は、右側にあった画像を左側に移動して、右側に新しい画像を表示するものとする



ヒント

- ページに表示される1番目の画像は`document.images[0]`なので、2番目の画像は`document.images[1]`となる

問題10-2 この章の課題の「スライドショー」で画像を1度に2枚ずつ表示するようにせよ。前に表示した画像は表示しないで、2枚とも新しい画像に変えるものとする



問題10-3 問題10-2を30枚の画像を表示できるように変更せよ

第4章の問題4-12で説明した`toString`と`padStart`を使ってください。

バリエーション

- 一度に表示する画像の枚数を3枚、4枚、...と増やしてみよ

問題10-4 ページを表示すると3秒後にランダムに選んでニュースサイトに移動するページを作れ。移動する前にダイアログボックスを使って移動するかどうか確認するものとする

ヒント

- **window.location**がもっているプロパティを使うと、任意のURLに移動できる
- **confirm**を使うと [OK] か [キャンセル] かを尋ねるダイアログボックスを表示する。ユーザーが [OK] を選択すると**true**が戻る、[キャンセル] だと**false**が戻る

```
<body>

<h1>ランダムなニュースサイトへ移動（日本語識別子バージョン）</h1>

<script>
  "use strict";
  // 配列sitesにニュース系サイトのURL（「https://」のあと）を記憶
  const サイトの配列 = ["asahi.com", "mainichi.jp", "yomiuri.co.jp",
    "nikkei.com", "news.yahoo.co.jp", "sanspo.com",
    "nikkansports.com", "gunosy.com", "oricon.co.jp"];
  const i = ランダムな整数を取得(サイトの配列.length-1); // 添字の整数を得る
  // サイトの配列.lengthで大きさ。配列の添字は0からになるので、大きさから1引く
  const url = "https://" + サイトの配列[i]; // +は連結（くっつける）
  const ok = confirm(`OKを押すと ${url} へ移動します`); // ダイアログを表示
  if (ok) { // [OK]を選択されたらokの値がtrue（真）になる
    document.write(`3秒後に ${url} へ移動します。`);
    setTimeout(()=> ●●● = url, 3*1000) // ←●●●を変える
  }
  else { // [キャンセル] を選択された場合
    document.write("キャンセルしました。"); // 移動はしない
  }

  function ランダムな整数を取得(n) { // 0以上n以下の整数をランダムに得る
    let x = Math.random(); // 0 ≤ x < 1
    x = x * (n+1);          // 0 ≤ x < n+1
    x = Math.floor(x);      // 0 ≤ x ≤ n (xは整数)。Math.floorで整数に切り下げ
    return x;
  }
</script>
```

問題10-5 ページを表示すると3秒後に以下の条件を満たすニュース系サイトに移動するページを作れ。移動する前にダイアログボックスを使って確認するものとする

- 午前0時から午前9時直前まで -- **https://asahi.com**
- 午前9時から午後5時直前まで -- **https://nikkei.com**
- 午後5時すぎから午後9時直前まで -- **https://oricon.co.jp**
- 午後9時から翌日午前0時直前まで -- **https://sanspo.com**

ヒント

- 第9章の練習問題9-4（時間によってあいさつを変える問題）を参考に
- システムのタイムゾーンを変えると、別の時刻でうまくいくか試せます

バリエーション

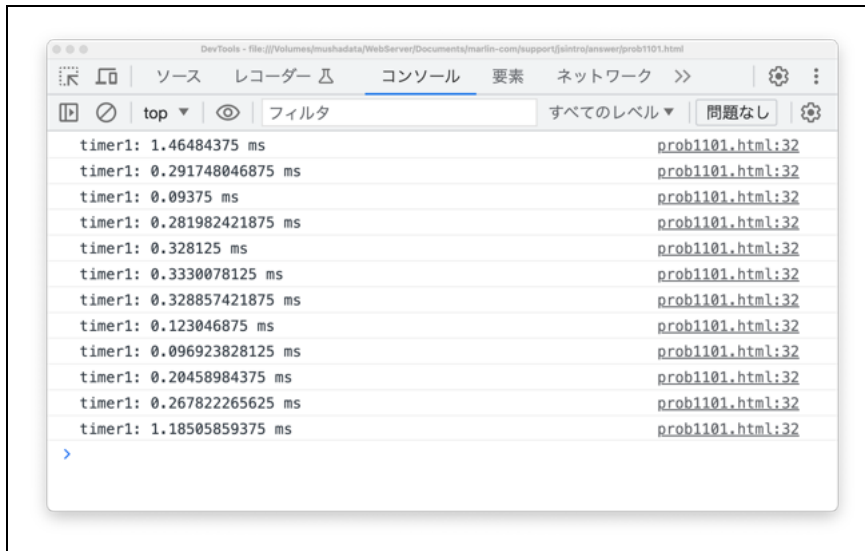
- それぞれの時間帯で、複数の候補を設定して、ランダムに移動するようにせよ
 - 午前0時から午前9時直前まで -- テレビ局サイト
 - 午前9時から午後5時ちょうどまで -- 一般新聞社サイト
 - 午後5時すぎから午後9時直前まで -- スポーツ新聞サイト
 - 午後9時から翌日午前0時直前まで -- エンタメ系サイト

第11章 11.5 練習問題

問題11-1 この章の課題（`example/ch1101.html`）のコードに時間を計測するコードを挿入し、カウントダウンの処理（数字などの表示）にどの程度の時間がかかっているかを計測せよ【写経】

```
9 <body>
10   <div style="width: 320px; text-align: center; font-size: 36pt;">
11     ロケット発射<br>
12     
13     <div id="counterArea"></div> <!-- ここに数字等を書く。最初は空白 -->
14   </div>
15
16 <script>
17   "use strict";
18   let count = 10;
19   let timerID = setInterval(countDown, 1000);
20
21   function countDown() {
22     console.time("timer1") // 【計測開始】
23     if (0 <= count) { // 「カウンタ」の値が0以上
24       document.getElementById("counterArea").innerHTML = count;
25       count--; // 次に備える
26     }
27     else { // countの値が 0 より小さくなったのでカウントダウン終了
28       clearInterval(timerID); // タイマー停止
29       document.getElementById("counterArea").innerHTML = "発射！";
30       document.images[0].src = "pictures/rocket2.png"; // 画像を交換
31     }
32     console.timeEnd("timer1") // 【計測終了】
33   }
34 </script>
```

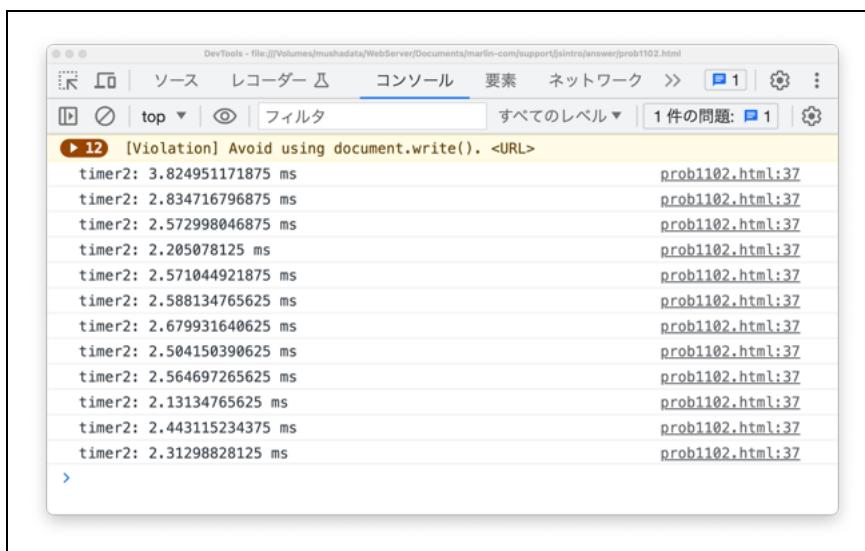
【計測開始】から【計測終了】までにかかった時間が、たとえば次のようにコンソールに表示されます（Chromeのコンソールの表示です）。



問題11-2 第6章の課題（example/ch0601.html）のコードに時間を計測するコードを挿入し、カウンタダウンの処理（数字などの表示）にどの程度の時間がかかっているかを計測せよ。問題11-1の結果と比較せよ。

- ch0601.htmlのコードの関数「画面書き換え」の先頭に**console.time**を、最後に**console.timeEnd**を挿入する

筆者のパソコンでの実行結果(Chromeのコンソールの表示)は、次のようになりました。

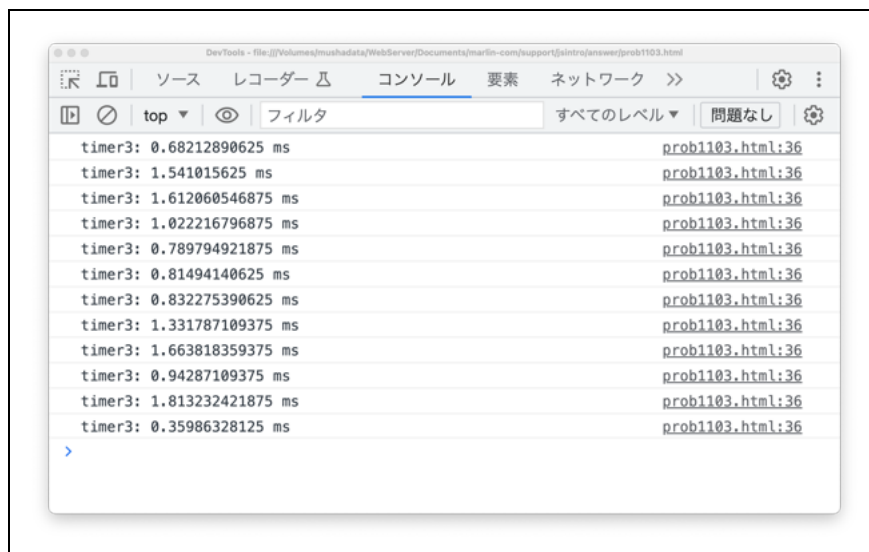


11-1では最初と最後の行を除き0.4ms以下なのに対して、11-2の結果ではすべて2msを超えています。11-2では、全画面を書き直しているため時間がかかっているようです。

Chromeで実行すると、「Avoid using document.write()」(document.write()の使用を避けよ)という警告が表示されます。上のコラム「document.writeについて」で説明したように「document.write()」は効率がよくないので、使わないようにせよ」と警告してきます。

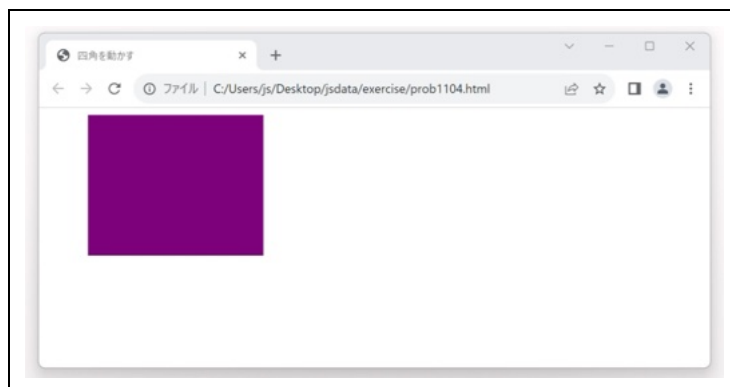
問題11-3 問題11-2のコードから`document.write`を取り除き、`document.getElementById`を使って書き出して、結果を比較せよ。

筆者のパソコンでの実行結果(Chromeのコンソールの表示)は、次のようになりました。



この結果を問題11-2の結果と比べてみると、パソコンで行っている他の処理などによっても状況が変わりますので、一概には言えませんが、問題11-3のほうが速くなっているようです。

問題11-4 教科書本文の「11.3 スタイルとアニメーション」の「位置の変化」で動かした長方形を、CSSの背景色 (`background-color`) を使って描き、同じように左から右に動かせ



* `<div>...</div>`で領域を定義して、幅と高さを設定し、その背景色を指定します。たとえば、次のようなコードを書きます (`exercise/prob1104.html`)。

```

7   <style>
8     #rectangle { /*「長方形」の領域 */
9       background-color: purple; /* 背景色の指定 */
10      width: 200px; /* 幅 */
11      height: 160px; /* 高さ */
12    }
13  </style>
14 </head>
15 <body>
16   <div id="rectangle"> <!-- ここに長方形が描かれる。上のスタイル指定参照 -->
17   </div>

```

16行目で、**div**領域に**rectangle**というidをふっています。この領域のスタイルは8行目から12行目で指定されています。style指定の各要素を説明しましょう。

- **background-color: purple** —— 背景色の指定。紫色（purple）。**red**、**white**、**blue**など英語名による指定のほか、**#FE33FF**などのような「16進数」の数値を使って指定することもできます。詳しくは次のページを参照してください —— <https://musha.com/scjs/?ln=color>
- **width: 200px** —— 幅200ピクセル
- **height: 160px** —— 高さ160ピクセル

これで準備ができたので、**<div id="rectangle">...</div>**で描いた長方形の左端からのマージンをJavaScriptを使って変化させます。

変化させる方法は、画像の場合とほぼ同じです。**document.getElementById**を使って長方形を表す**<div>...</div>**のオブジェクトを取得して、マージンを変化させます。

たとえば次のようなコードになります。

```

const タイマーID = setInterval(四角を動かす, 50); // 50ミリ秒ごとに実行
let カウンタ = 0;

function 四角を動かす() { // setIntervalで呼び出される関数
  カウンタ++;
  const 四角 = document.getElementById("rectangle");
  四角.style.marginLeft = `${カウンタ}px`;
  // 左マージン（ウィンドウ左端から四角までの距離）を指定
  if (800 < カウンタ) { // 800ピクセルを超えて移動したら終了
    clearInterval(タイマーID);
  }
}

```

問題11-5 問題11-4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

問題11-6 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）

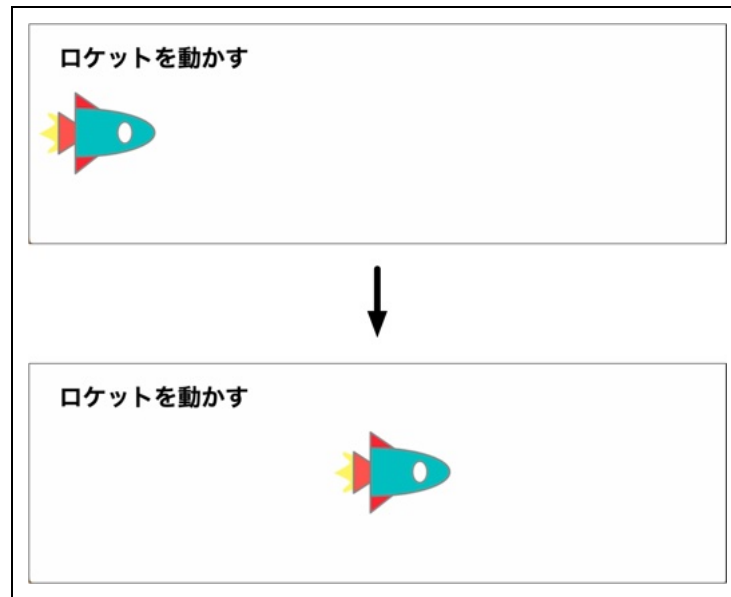
- 右側に来たら、方向を変える → marginの値を減らしていく

コード例

```
25 "use strict";
26 const 継続時間 = 20; // 「継続時間」実行したら停止。ずっと動き続けないように
27 const 更新間隔 = 10; // setIntervalの第2引数（ミリ秒単位）
28 const 右方向への移動 = 1;
29 const 左方向への移動 = 2;
30 const 左マージンの最大値 = 800; // px単位
31 const 一回の移動距離 = 3; // 1回に動く距離（px単位）
32
33 const 四角 = document.getElementById("rectangle");
34 let 方向 = 右方向への移動;
35 let 左マージン = 0;
36
37 const タイマーID = setInterval(四角を動かす, 10);
38 setTimeout(()=>clearInterval(タイマーID), 継続時間*1000);
39 // 継続時間（20）秒、たったらやめる
40
41 function 四角を動かす() {
42   if (方向 === 右方向への移動) { // 右方向の場合
43     左マージン += 一回の移動距離; // 右に移動することになる
44     if (左マージン < 左マージンの最大値) { // まだ一番右に到達してい
45       四角.style.marginLeft = `${左マージン}px`;
46     }
47   } else { // 行き過ぎてしまうので方向を逆にする
48     方向 = 左方向への移動;
49   }
50 }
51 else { // 左方向への移動の場合
52   左マージン -= 一回の移動距離; // 左に移動することになる
53   if (0 < 左マージン) { // まだ左端に到達していない
54     四角.style.marginLeft = `${左マージン}px`;
55   }
56   else { // 左端に達してしまうので方向を逆にする
57     方向 = 右方向への移動;
58   }
59 }
60 }
```

exercise/prob1106b.htmlに別解があります。コメントを付けておきましたので、解読してみてください。

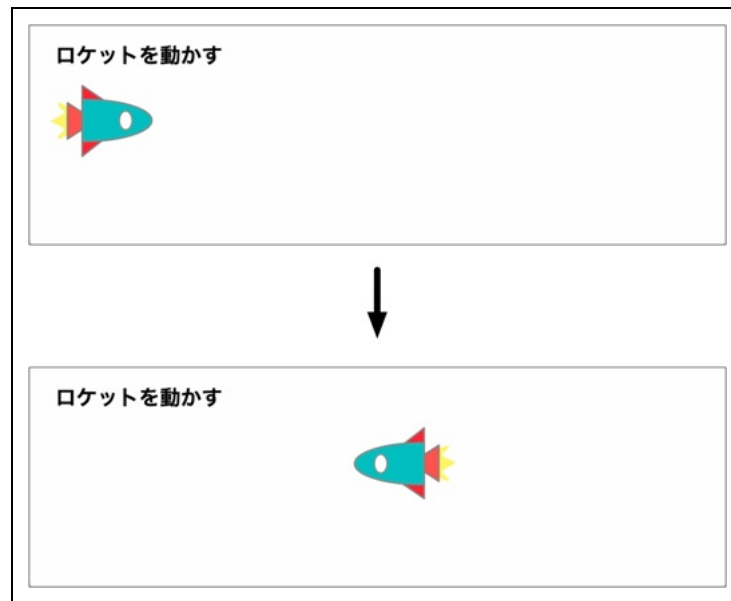
問題11-7 画面上にロケットの画像（pictures/rocket3.png）を表示し、左から右に動かすプログラムを作れ【写経】



【exercise/prob1107.html】

```
17 <div id="rocket">
18   
19 </div>
20
21 <script>
22   "use strict";
23   const ロケット = document.getElementById("rocket");
24   // 画像タグがここより上にないとオブジェクトを取得できないので、<img>は<script>よりも上を書く
25
26   const 動かす幅 = 400; // px単位
27   let カウント = 0;
28   const タイマーID = setInterval(ロケットを動かす, 10);
29
30   function ロケットを動かす() {
31     カウント++;
32     if (動かす幅 < カウント) {
33       clearInterval(タイマーID);
34     }
35     ロケット.style.margin = `0 0 0 ${カウント}px`; // 上, 右, 下, 左の順にマージン指定
36     // ロケット.style.marginLeft = `${カウント}px`; // ←これでもOK
37   }
38 </script>
```

問題11-8 ロケットを左から右、右から左と交互に動かすプログラムを作れ。折り返す場所は、ロケットが画面からはみ出ない適当な場所にせよ（`pictures/rocket3.png`と`pictures/rocket4.png`の画像を使う）



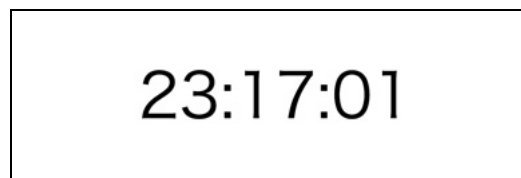
問題11-9 問題11-8で表示したロケットを、ウィンドウの右端より少し左で折り返すようにせよ

- `window.innerWidth` でドキュメント領域の幅（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

第12章の問題12-3以降で、イベント処理に関してさらに改良します。

問題11-10 デジタル時計を表示せよ（24時間制）

- 問題9-4で説明されているDateオブジェクトを利用する
- 24時間制の時、分、秒を刻々と表示する
- 一桁の数字の前には0を入れて二桁にする



ヒント

- `setInterval`を使って短時間で繰り返し更新する

問題11-11 デジタル時計を表示せよ（午前、午後）

- 12時間制の時、分、秒を刻々と表示する
- 午前あるいは午後を頭に入れる

- 一桁の数字の前には0を入れて二桁にする

午後11:17:44

ヒント

- 午前と午後で処理を分ける

問題11-12 2021年に開催された東京オリンピックの開会式の日（7月23日）からの日数を表示するプログラムを作成せよ

2度目の東京オリンピックの開会式の日（2021年7月23日）から **840日** たちました。

- `new Date()` で日付を指定することもできる
- `Math.floor` は整数に切り下げ —— `Math.floor(1340.6) -> 1340`

```
const 開催日午前零時 = new Date(2021, 6, 23);  
// 年、月、日を指定（6で7月を表すことに注意）  
// 1970年1月1日午前0時0分0秒（UTC）からのミリ秒数で、現在時刻を表す  
  
const 現在 = new Date();  
const 差_ミリ秒 = 開催日午前零時 - 現在;  
const 差_日_小数 = 差_ミリ秒 / 1000 / 60 / 60 / 24; // 1000ミリ秒が1秒, 60秒が1分, 60分が1時間, ...  
const 差_日_整数 = Math.floor(差_日_小数) + 1;  
...
```

第12章 12.10 練習問題

イベント処理を身につけると、ムービーや画像を使って楽しいことがいろいろできるようになります。皆さん自身で、いろいろな「バリエーション」を考えてみてください。

問題12-1 ロケットの画像をクリックしたら「クリックしました！」とダイアログボックスに表示するプログラムを作れ

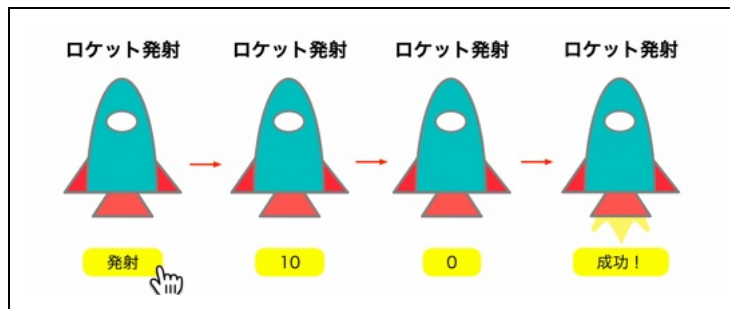


コード例 (exercise/prob1201.html)

```
...
7 <style>
8   #rocket { /* id="rocket" についての指定 */
9     width: 200px; /* ロケットの画像の大きさを指定 */
10    cursor: pointer; /* 上に来るときマウスポインタを指の形にする */
11  }
12 </style>
13 </head>
14
15 <body>
16   <div>
17     <h1>画像をクリック</h1>
18     
19   </div>
20
21   <script type="text/javascript">
22     "use strict";
23     const ロケットオブジェクト = document.getElementById("rocket");
24
25     ロケットオブジェクト.addEventListener("click", () => {
26       ●● // ← ●●を変更
27     });
28   </script>
29 </body>
30 </html>
```

問題12-2 ロケットの画像の下に「発射」ボタンを置き、ボタンを押したらカウントダウンを開始して、0 になったらロケットの画像を変えて、発射したように見せよ【写経】

- 画像はpictures/rocket1.pngとpictures/rocket2.png
- 画像の領域とカウントダウンの領域を設ける
- 2つの領域に名前を付けて、innerHTML（あるいはsrc）を変更



コード例 (exercise/prob1202.html)

```
...
<style>
  img { width: 200px; } /* ロケットの画像の大きさを指定 */
  #button {
    background-color: yellow;
    width: 200px;
    margin: 0;
    padding: 10px 40px; /* 文字と枠の間の距離。上下10px、左右40px */
    font-size: 20pt; /* 文字の大きさ20ポイント */
    cursor: pointer; /* ポインタを手の形にする */
    border-radius: 15px; /* 角を丸くする。15pxの半径 */
  }
</style>
</head>
<body>
  <div style="text-align: center;">
    <h1>ロケット発射</h1>
     <!-- 画像 --><br>
    <span id="button">発射</span> <!-- 数字に変わる -->
  </div>

  <script>
    "use strict";
    let カウント = 10;
    let タイマーID = -1; // 下のif文で使う
    const ボタンオブジェクト = document.getElementById("button");

    function カウントダウン() { // 1秒ごとに呼ばれる関数
      if (カウント >= 0) {
        ボタンオブジェクト.innerHTML = カウント; //数字変更
      }
      else { // 0より小さくなったら
        ボタンオブジェクト.innerHTML = "成功!";
        document.images[0].src = "pictures/rocket2.png"; // 画像を変える
        clearInterval(タイマーID); // 繰り返しタイマーを止める
      }
      カウント--; // ひとつ減らす (マイナス「-」2つ)
    }

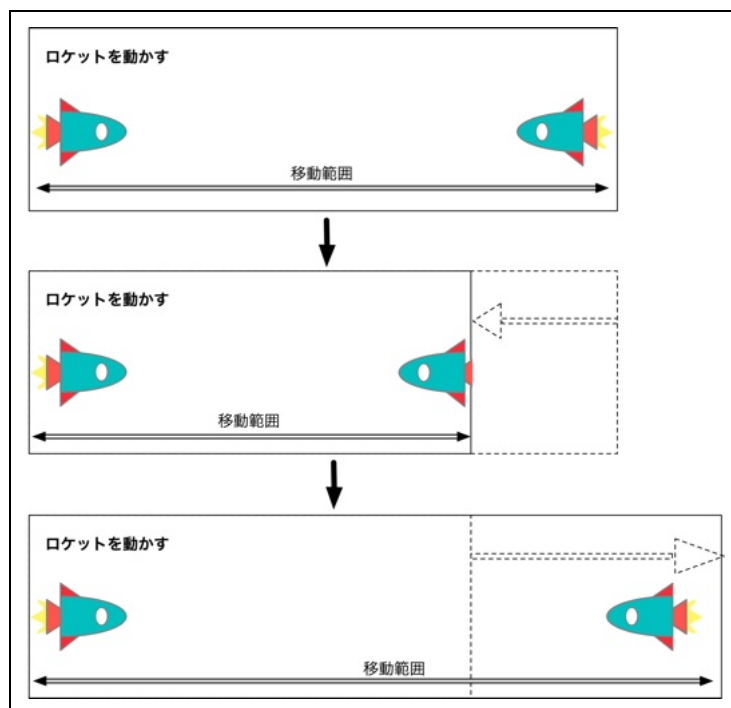
    ボタンオブジェクト.addEventListener("click", () => {
      if (タイマーID === -1) {
        ボタンオブジェクト.innerHTML = カウント;
        // ↑上の行を入れたほうがすぐに反応したように見える。
        // 上の行をコメントにしてみると、1秒たってから始まることになる
        カウント--;
        タイマーID = setInterval(カウントダウン, 1000); // 1秒ごとに呼び出す
      }
      // タイマーID === -1の条件がないとボタンをクリックするたびに
      // 別のタイマーが始まり、1秒間に何度もカウントダウンが呼ばれることになる
      // カウントダウンが速くなってしまふ
    })
  </script>
</body>
</html>
```

```
});
</script>
</body>
</html>
```

問題12-3 ロケットが左から右、右から左と交互に動くプログラムを作れ。ただし、ウィンドウの大きさを変えても右端付近で折り返すようにせよ。また、30秒経過したらロケットを停止せよ（問題11-9を参考に）

- ウィンドウの大きさを変えるとwindowオブジェクトにresizeイベントが発生するので、このイベントが起きたときの処理を記述すればよい

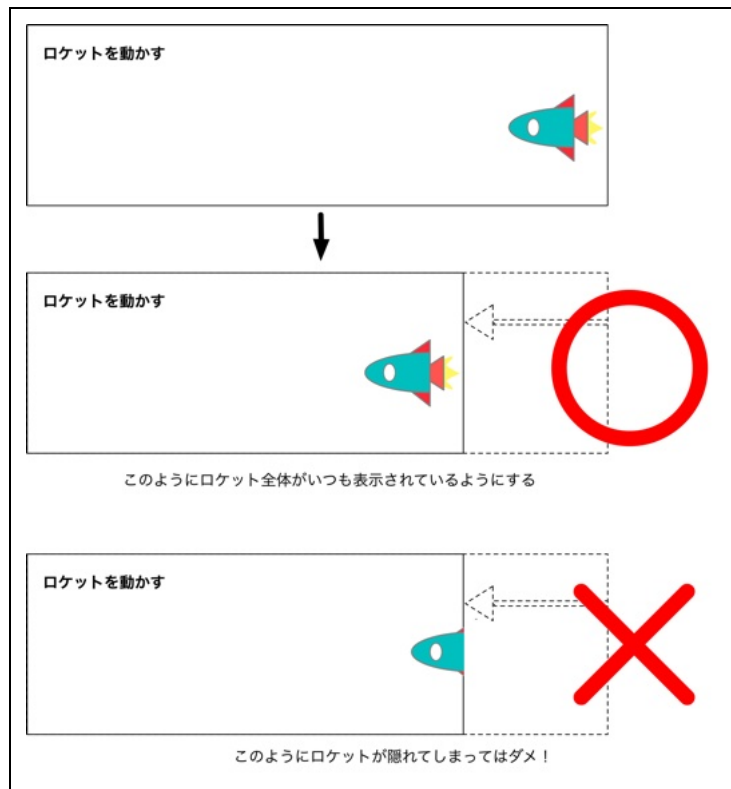
```
window.addEventListener("resize", () => {
  // ウィンドウのサイズが変わったときの処理をここに書く
});
```



問題12-4 上の問題で、最後にロケットを止める際に、30秒経過後に左端まで戻ったときに停止するようにせよ。ロケットの向きは最初と同じように右向きにする

問題12-5 上の問題で、ロケットが右端のほうにあるときに、ウィンドウ幅を左に縮めても、ロケットが隠れてしまわないようにせよ

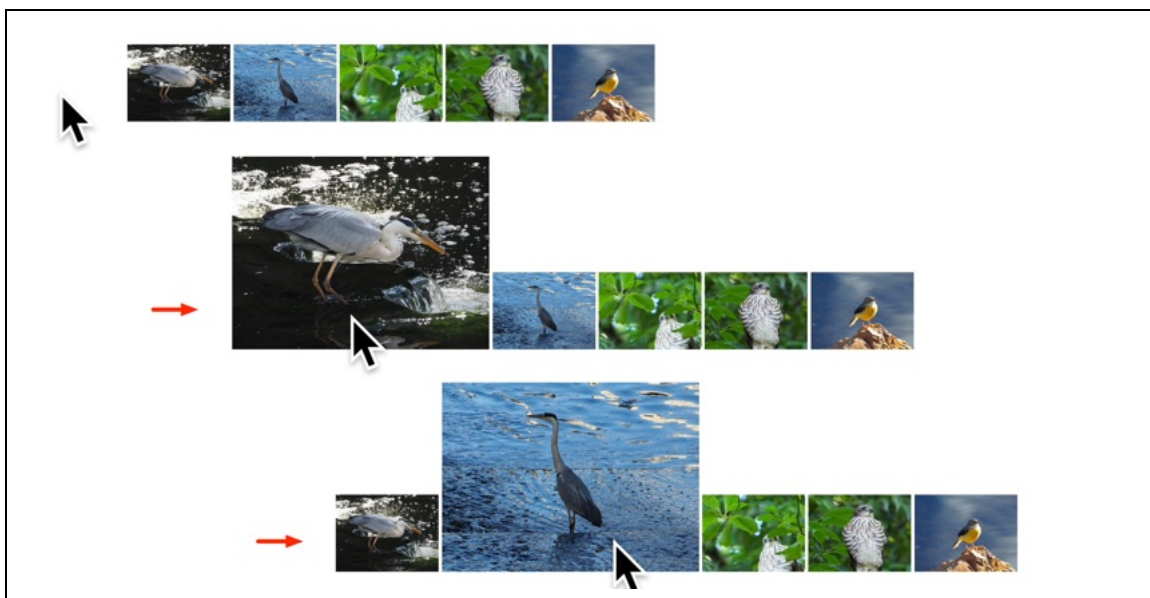
上の問題(12-4)の解答例のコードだと、素早く左にドラッグしてウィンドウ幅を小さくすると、ロケットが隠れてしまいます。ロケットはいつも全体が表示されているようにしてください。



ヒント

- `resize`の際にウィンドウの幅が変わるところで、ロケットの左マージンもチェックしてロケットがウィンドウからはみ出さないようにする

問題12-6 画像を5つ横に並べ、マウスが上にきた画像だけを拡大するようなページを作れ。画像の下辺を揃えるように工夫せよ (★難問★)



- 画像の大きさを変えるコードの例

```
// 「id が "pict0"の画像」の大きさ (width) を変えるには...
 ←画像にidを振っておく
...
```

```
<script>
...
let 画像 = document.getElementById("pict0");
画像.style.width = "400px"; // 画像の大きさを変更
...
</script>
```

- マウスが画像の上に来た -- x.addEventListener("mouseover", () => {...});
- マウスが画像の外に出た -- x.addEventListener("mouseout", () => {...});

問題12-7 上の問題で、画像を拡大するときに少しずつスムーズに拡大するようにせよ（上の問題の解答例では、画像が突然拡大されるが、少しずつ拡大されるようにする）

ヒント

拡大する画像のCSSに、**トランジション (transition)**を指定します。たとえば次のように指定することで、コメントにあるような効果が得られます。

```
transition-property: all; /* 拡大・縮小時に幅や高さなどすべてのプロパティが変化 */
transition-duration: 200ms; /* 開始から終了までの時間 */
transition-delay: 0s; /* 開始するまでの時間 */
transition-timing-function: ease-in; /* 開始時はゆっくり、終了時は早く変化 */
```

また、これらを次のようにまとめて指定もできます。

```
transition: all 200ms 0s ease-in; /* 上の4行と同じ効果 */
```

画像のCSSに上のコードを加えれば、スムーズな動きになります。

なお、transitionの詳細は次のページなどを参照してください。

- 株式会社アーティスのブログ記事 — <https://musha.com/scjs?ln=1203>
- mdnのページ — <https://musha.com/scjs?ln=1204>

問題12-8 氏名を入力して「検索」ボタンを押すと、その人の電話番号を表示する「フォーム」を使ったページを作成せよ。登録されていない氏名を入力した場合は、「わかりません」と表示する【写経】

電話帳

氏名：

電話番号：1268-22-4321

コードに説明が書いてありますので、よく読んでください。重要なのは次の2箇所です。

- 「フォーム」を「送信 (submit)」すると、submitというイベントが発生します。したがって、formのオブジェクトのaddEventListenerでsubmitを捕捉します（40行目）

- フォームのinput領域の値はvalueというプロパティを見ればわかります (52行目)

```
17 <body>
18 <h2>電話帳</h2>
19 <!-- **ex1401-bekkai.html に別解あり** -->
20 <form id="form1">
21   氏名:<input type="text" id="name">
22   <button type="submit" value="search">検索</button><br>
23   電話番号:<span id="phoneNumber"></span>
24 </form>
25
26 <script>
27 "use strict";
28
29 //電話帳の定義 オブジェクトのプロパティを使って記憶する
30 const 電話帳 = {
31   "東野ナカ": "160-8811-8888", "平山綾香": "190-3344-3333", "白木渚": "143-334-2222",
32   "武田信玄": "1268-22-4321", "上杉謙信": "13-3311-3421", "真田幸村": "1268-24-3311",
33   "松ひく子": "160-3312-2212"
34 };
35 // 次のようにしてもよい（プロパティ名は "... " でくくなくてもよい）。
36 // 東野ナカ: "160-8811-8888", 平山綾香: "190-3344-3333", ...
37
38
39 // submitイベントが起こったときに関数「電話番号を表示」を呼び出す
40 document.getElementById("form1").addEventListener("submit", 電話番号を表示);
41 /* *注意*
42   document.getElementByIdに指定するIDは、このコードより上で定義されていないとエラーになる
43   (ただし、少し「細工」をすればこの制限をなくすこともできる。たとえばonloadイベントを使う)
44 */
45
46 function 電話番号を表示(event) {
47   event.preventDefault(); // 再読込しないようにする
48   // フォームは、submitするとデフォルトでは（何も指定しないと）指定のURLに移動してしまう。
49   // URLが指定されていない場合は同じページを再読み込みする。
50   // すると、名前も電話番号も消えてしまう。
51   // 上を書いておくとデフォルトの動作をしない -> 再読み込みしない
52   const 氏名 = document.getElementById("name").value;
53   // nameというidのinput領域に入力された値を取得
54   // console.log(氏名); // デバッグ用
55   const 電話番号 = 電話番号を検索(氏名);
56   document.getElementById("phoneNumber").innerHTML = 電話番号;
57 }
58
59 function 電話番号を検索(氏名) {
60   if (電話帳[氏名]) { // 電話帳に氏名が登録されていれば
61     return 電話帳[氏名];
62   }
63   return "わかりません";
64 }
65 </script>
66 </body>
```

問題12-9 問題9-1と同じデータ（英和辞書）を用いて、英単語を入力して【検索】ボタンを押すと、日本語訳を表示するフォームを使ったページを作成せよ

英和辞書

英単語：

日本語の意味：図書館

ヒント

- 構造はすぐ上の問題と同じ。データを変えればよい。関数名は変えなくても動作するが、関数の名前と処理内容が一致しなくなりわかりにくいプログラムになってしまうので、関数名も変更する

問題12-10 問題12-9と同じ英和辞書のデータを用いて、日本語の単語を入力して【検索】ボタンを押すと、対応する意味をもつ英単語を表示するフォームを使ったページを作成せよ（問題9-3で作った和英辞書を使う）

和英辞書

日本語の単語：

英単語： library

問題12-11 上の問題を組み合わせて、英単語を入れたときは日本語の意味を、日本語の単語を入れたときは対応する英単語を表示するページを作成せよ

英和・和英辞書

英語あるいは日本語の単語：

対応する意味をもつ日本語あるいは英語： elephant

英和・和英辞書

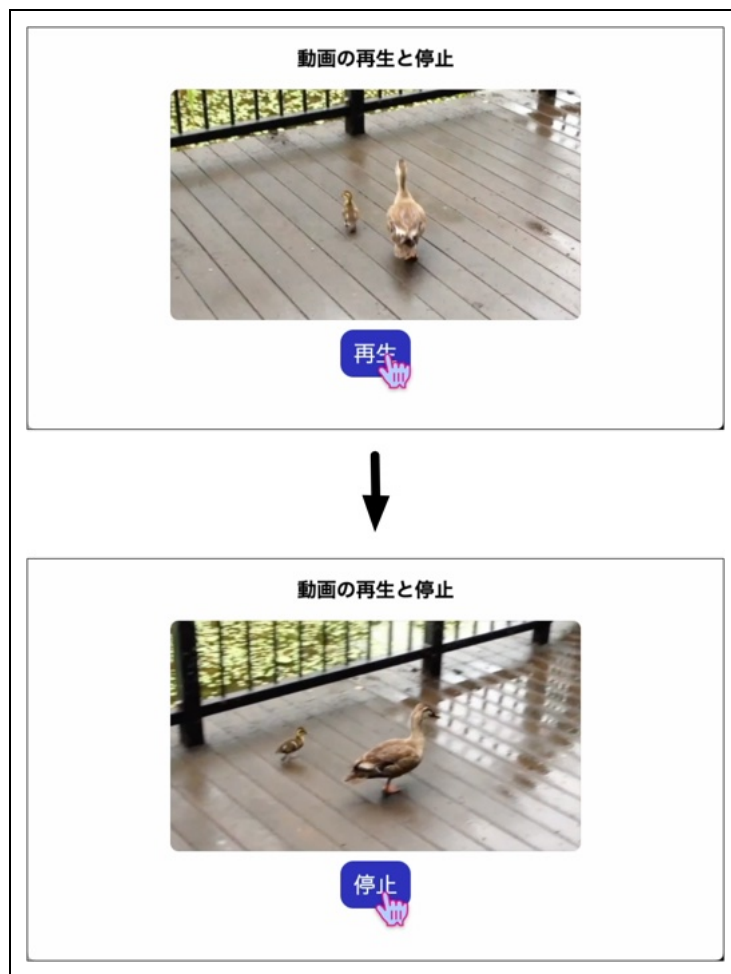
英語あるいは日本語の単語：

対応する意味をもつ日本語あるいは英語： 図書館

問題12-12 下図のような動画を表示するページを作れ【写経】

- [再生] ボタンを押すと再生を開始し、ボタンが[停止] になる
- 再生中に[停止] ボタンを押すと再生を停止し、ボタンが[再生] になる

第9章の「9.3 動画オブジェクト」の説明も参照してください。



```
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>動画の再生と停止</title>
7   <style>
8     .movie {
9       width: 60%; /* 幅60%で表示 */
10      border-radius: 10px; /* 境界線 (border) の角を丸く。半径 (radius) 10px */
11    }
12    .controller {
13      margin: 1rem;
14      text-align: center;
15      font-size: 20pt;
16    }
17    .button {
18      width: 30%;
19      color: white;
20      background-color: #0032BF;
21      padding: 1rem;
22      border-radius: 1rem;
23      cursor: pointer; /* マウスポインタの手の形にする */
24    }
25  </style>
```

```

26 </head>
27
28 <body style="text-align: center;">
29   <h2>動画の再生と停止</h2>
30
31   <div>
32     <video class="movie" src="movies/no00.mp4" poster="movies/no00.jpg"
33       playsinline loop> <!-- 再生を繰り返す -->
34   </video>
35 </div>
36
37 <div class="controller">
38   <span class="button" id="startStop">再生</span>
39 </div>
40
41 <script>
42   "use strict";
43   const movies = document.getElementsByClassName("movie");
44   // CSSのclass名が"movie" のものを全部集める
45   // moviesはHTMLCollectionというほぼ配列と同じように扱えるものになる
46   // この例では1つしかないので、getElementByIdを用いてもよいが、
47   // あとの問題で複数の動画を扱うので、拡張できるようにこうしておく
48
49   const startStopButton = document.getElementById("startStop");
50
51   startStopButton.addEventListener("click", () => { // ボタンをクリックしたときの処理
52     if (movies[0].paused) { // 再生中でないなら
53       movies[0].play();
54       startStopButton.innerHTML = "停止";
55     }
56     else { // 再生中のとき
57       movies[0].pause(); // 動画を停止
58       startStopButton.innerHTML = "再生";
59     }
60   });
61 </script>
62 </body>

```






問題12-13 問題12-12の動画ページに、下図に示すようなサウンドのオン（🔊）・オフ（🔇）を切り替えるボタンを付けよ（絵文字を利用）



問題12-14 問題12-13の動画ページに、下図に示すような「5秒戻る」「5秒進む」のボタンを追加せよ







問題12-15 4つの動画を順番に再生する、下記の条件を満たすページを作れ

- ページのロード時（最初に読み込んだとき）には、動画が4つ並び、その下に【停止】ボタンと現在サウンドがオフであることを示す  が表示される（下図参照）
- 動画はページのロード時に自動的に左から順番にn秒間ずつ再生され、最後まで再生すると最初の動画に戻る（nはプログラムの冒頭で定数で指定する）
- ページのロード時は、サウンドはミュート（消音）になっている
- 【停止】ボタンをクリック（タップ）すると、動画の再生を停止し、ボタンを【再生】に変更する
- 【再生】ボタンをクリック（タップ）すると、再度動画が順番に再生され、ボタンを【停止】に変更する。先ほど停止していた動画から再生を再開するものとする
-  ボタンをクリックするとサウンドが聞こえるようになり、ボタンは  に変わる
-  ボタンをクリックするとサウンドがミュートになり、ボタンは  に変わる



問題12-16 「この章の課題」の動画版を作成せよ

- 下図のようなレイアウトで、4つの動画を並べる
- マウスが上に来た場合にその動画を下の領域でより大きく再生する
- マウスが離れると、その動画を消去する
- 小さな画像右下にある  をクリックするとサウンドが聞こえるようになり、 に変わる
-  をクリックするとサウンドがミュートになり、 に変わる

