プログラミング技能検定試験

簡単なゲームや購入金額の計算などの身近なWebページの作成を通して、コンピューターはどのように命令すれば思った通りに動いてくれるのか、その仕組みを体感して学ぶことができます。

使用する言語

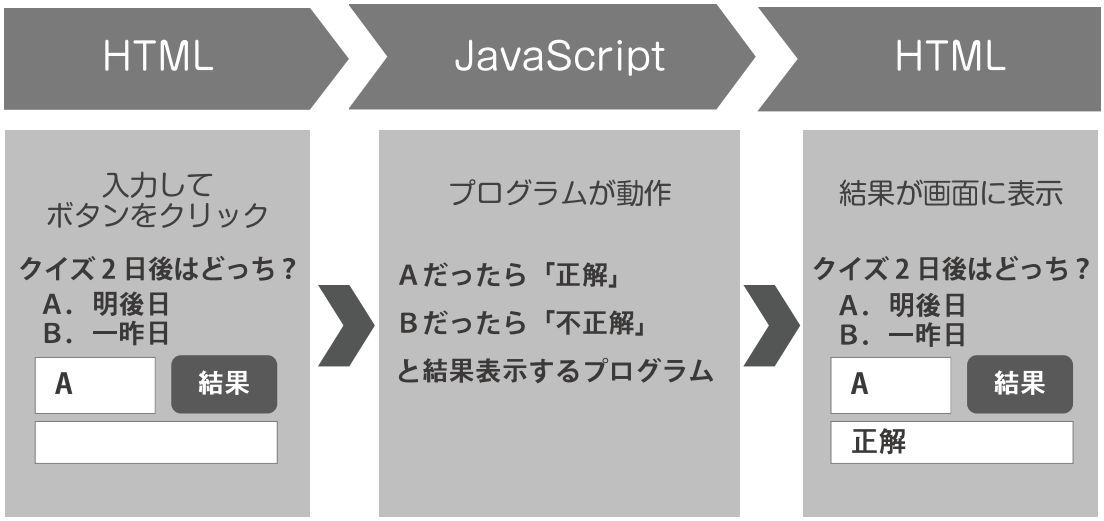
Webページ全体の構造（文字や画像などの情報）を記述する **HTML**

＋

Webページ上に動きを付ける **JavaScript**

どのようなWebページをつくるのか？

「入力」と「出力」を意識して下記のような流れの「動きのある」ページを作成します。



よくWebサイトで見掛ける「問い合わせフォーム」などのように、入力する欄やその内容を送信するボタンをHTMLで作成し、そこに入力されたものを受け取ってプログラムを組み立て、結果をまたHTMLに渡して表示させます。

何を使ってつくるのか？

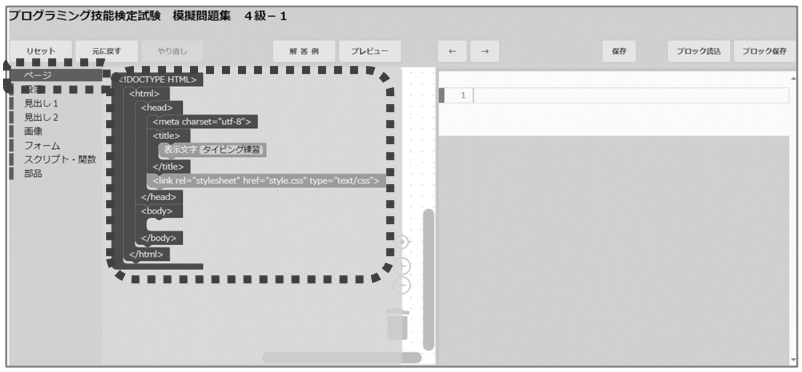
オンラインアプリケーションを使用します。

解答の流れ

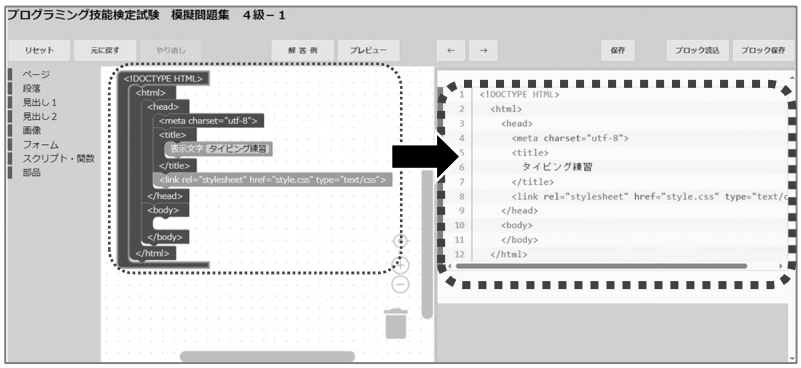
Web ページは、HTML・JavaScript で書かれたソースコードによって構成されます。※1
本試験（問題集）では、「ブロック」を用いてこれらを作成します。
問題文に従って、ブロックを配置し、その配置されたソースコードが右側のエリアに表示される仕組みになっています。「プレビュー」では、このソースコードをブラウザで表示した状態を確認することができます。

解答例）【ページ】のブロックセットを挿入しなさい。

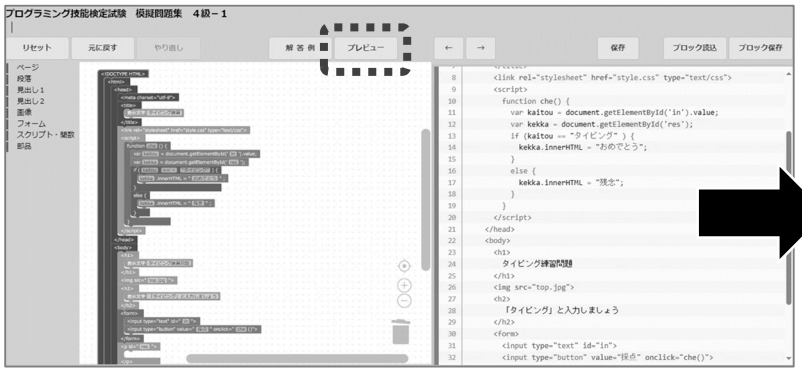
①　「ページ」をクリックし、中にあるブロックをクリック。



②　ドラッグして左側のエリアに配置。右側のエリアにソースコードが表示される。



問題文どおり処理が完了したら「プレビュー」をクリック。

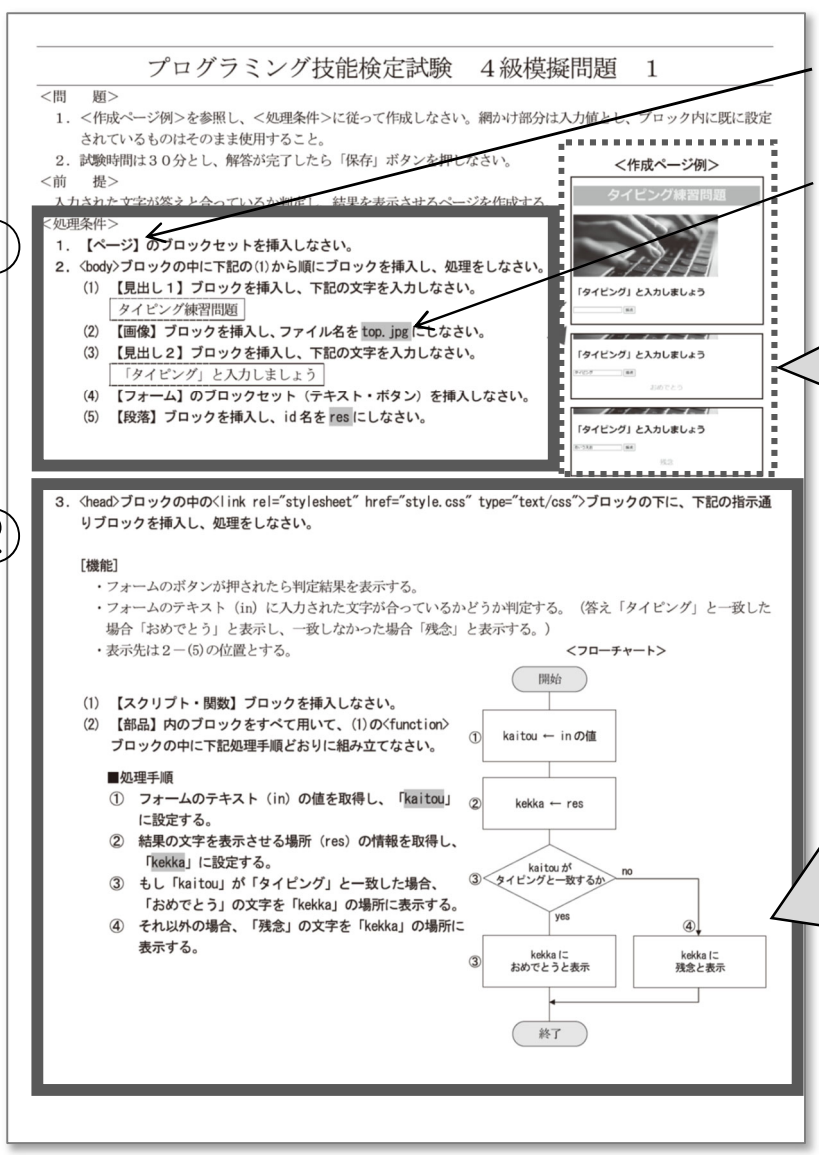


別タブに作成したソースコードを基にした Web ページが表示される。

ここで表示内容や動作を確認します。

※1：Web ページでは他に CSS というデザインを担当する言語も使用されます。本試験では、CSS は作成しませんが、既に準備された css ファイルへのリンクを行う出題はあります。ただし、その css ファイルへのリンクは【ページ】のブロックセットに既に配置されているため、あまり意識せずに解答できるようになっています。

問題の構成（４級）



＜問題＞…問題文の説明や試験時間、解答完了後の操作等について記載しています。

＜前提＞…どのような動きのページを作成するのか記載しています。

①表示部分を作成（HTML）

表示される文字や画像、入力する部分やボタン（フォーム）を作成します。処理条件の指示どおりにブロックを配置しましょう。アプリでは、フォームはブロックセットになっており、組み立ては必要ありませんが、入力部分がどのような仕組みになっているか確認をしておきましょう。

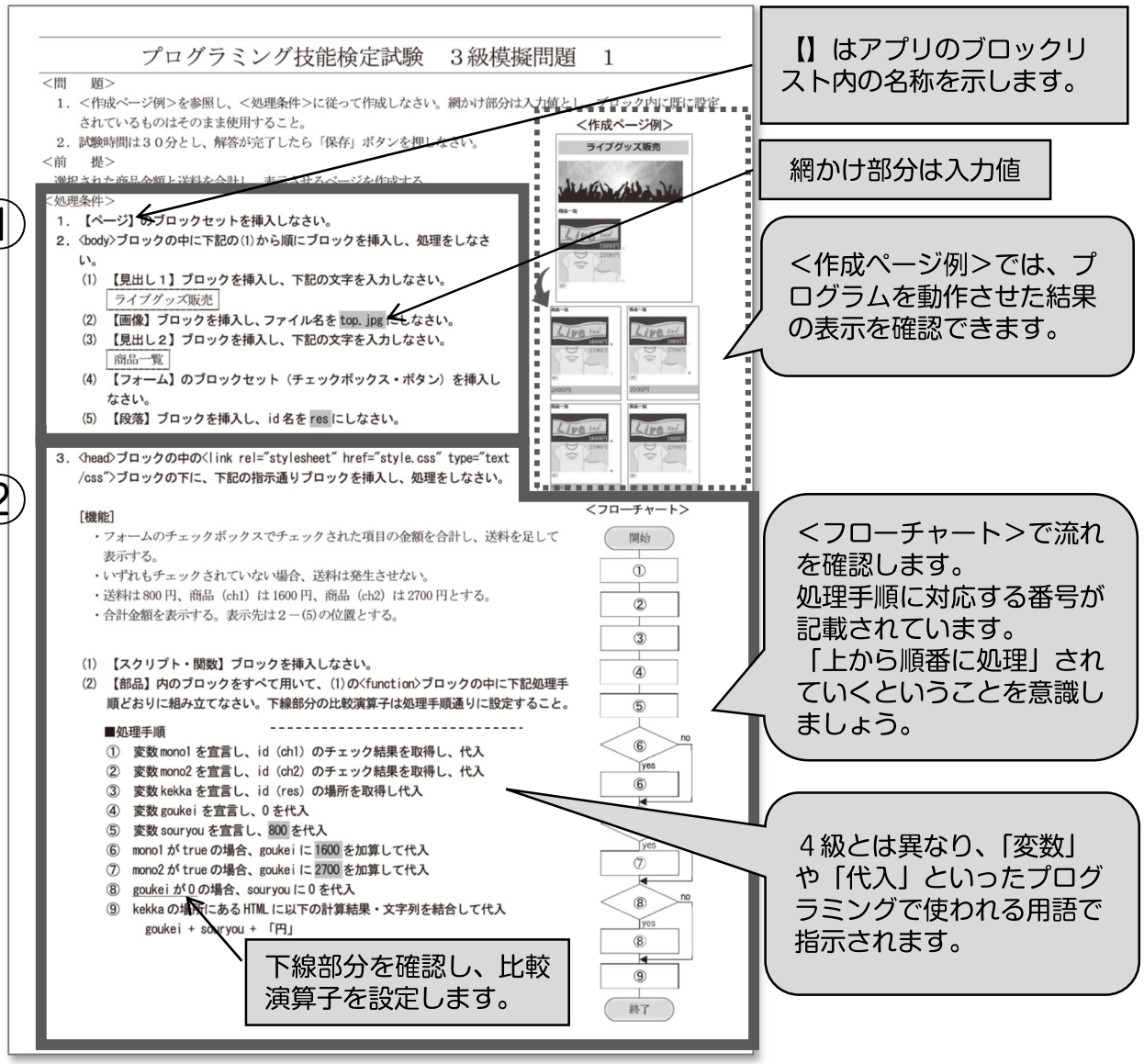
②動作部分を作成（JavaScript）

フォームのボタンが押された後の JavaScript の処理内容を示しています。

[機能]……ボタンが押された後にどのような動作をするのか、その結果をどこに表示するのかを記載しています。

■処理手順…フローチャートとともにその処理手順を確認し、順番通りにブロックを組み立てます。アプリには使用するブロックしか準備されていません。

問題の構成（３級）



＜問題＞…問題文の説明や試験時間、解答完了後の操作等について記載しています。

＜前提＞…どのような動きのページを作成するのか記載しています。

①表示部分を作成（HTML）

表示される文字や画像、入力する部分やボタン（フォーム）を作成します。処理条件の指示どおりにブロックを配置しましょう。アプリでは、フォームはブロックセットになっており、組み立ては必要ありませんが、入力部分がどのような仕組みになっているか確認をしておきましょう。

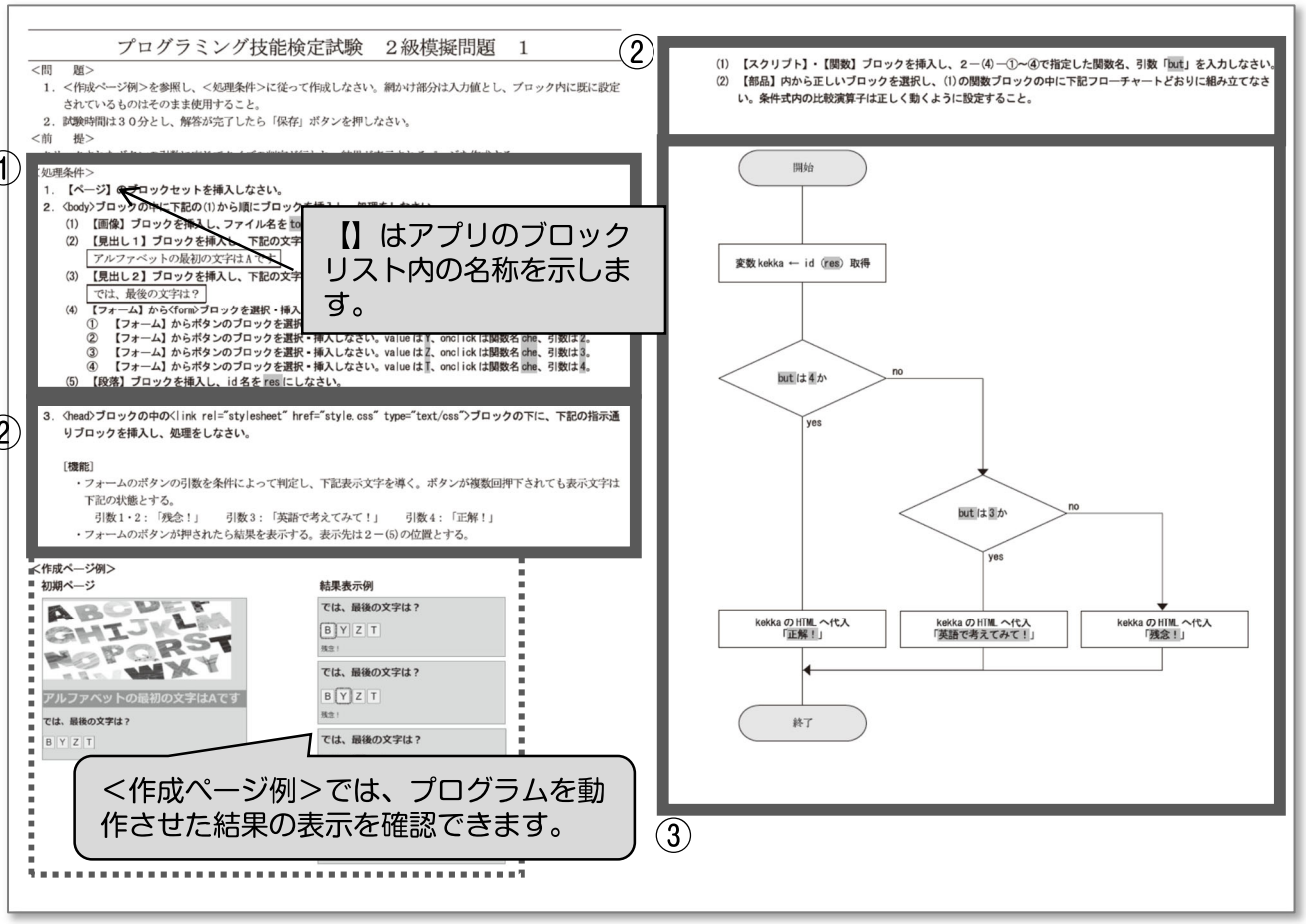
②動作部分を作成（JavaScript）

フォームのボタンが押された後のJavaScriptの処理内容を示しています。

[機能]……ボタンが押された後にどのような動作をするのか、その結果をどこに表示するのかを記載しています。

■処理手順…フローチャートとともにその処理手順を確認し、適したブロックを選択して組み立てます。アプリには使用するブロックしか準備されていません。

問題の構成（２級）



＜問題＞…試験時間や問題文の説明、解答完了後の操作等について記載しています。
網かけは入力値とします。

＜前提＞…どのような動きのページを作成するのか記載しています。

①表示部分を作成（HTML）

表示される文字や画像、入力する部分やボタン（フォーム）を作成します。処理条件の指示どおりにブロックを配置しましょう。フォームはプログラムのインプット部分になりますので、設定する際は、その先の JavaScript の動きを意識しながら行います。

②動作部分を作成（JavaScript）

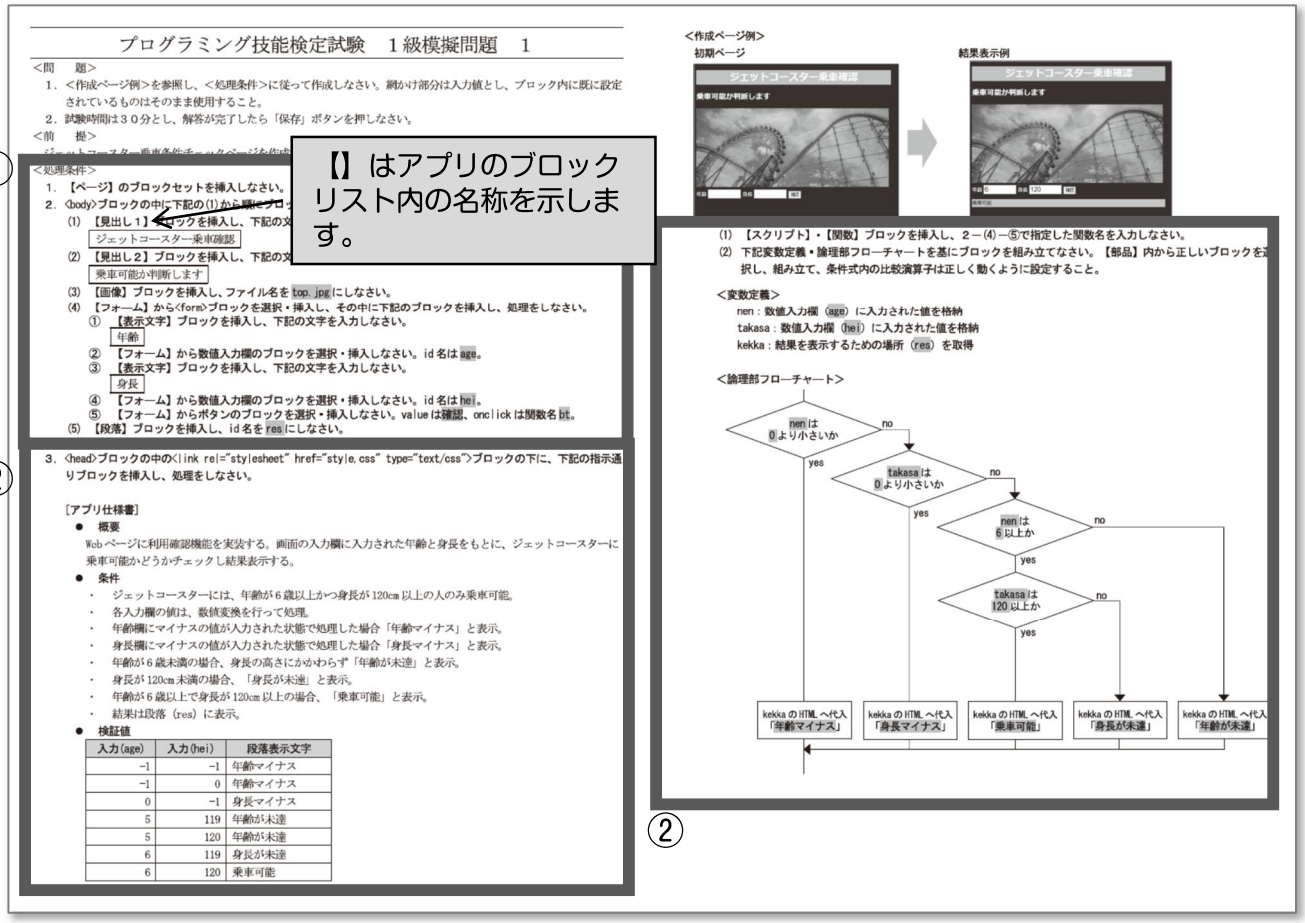
フォームのボタンが押された後の JavaScript の処理を組み立てます。フォームで設定した関数名を確認し、作成する JavaScript の関数名に設定します。そしてその関数の中の処理については、③のフローチャートで確認して組み立てます。

[機能]……ボタンが押された後にどのような動作をするのか、その結果をどこに表示するのかを記載しています。

③フローチャート

フォームのボタンが押された後の JavaScript の処理内容を示しています。
アプリの「部品」の中には、処理条件の指示通りにならない誤ったブロックも含まれているため、正しいものを選択して組み立てましょう。また、比較演算子も処理の内容にあわせて設定します。

問題の構成（１級）



＜問題＞・＜前提＞・①は２級と同様

②動作部分を作成（JavaScript）

フォームのボタンが押された後の JavaScript の処理を組み立てます。フォームで設定した関数名を確認し、JavaScript の関数に設定します。アプリの「部品」の中には、処理条件の指示通りにならない誤ったブロックも含まれているため、正しいものを選択して組み立てましょう。また、比較演算子も処理の内容にあわせて設定します。

[アプリ仕様書]

●**概要**……ボタンが押された後にどのような動作をするのか、その結果をどこに表示するのかを記載しています。

●**条件**……プログラムが実行される際に適用されるルールや要件を記載しています。具体的には、特定の状況や条件が満たされる場合にどのような動作をするか、またその結果表示内容を定義します。

●**検証値**……プログラムが正しく動作するかを確認するためのテストケースの一部を記載しています。作成後、「プレビュー」画面にて検証値を入力し、プログラムが条件を満たし、指示どおりの結果が得られるか検証しましょう。

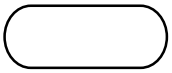
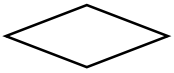
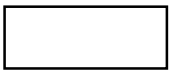
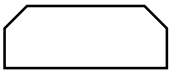
＜変数定義＞…プログラムで使用する変数、その設定を指示しています。

＜論理部フローチャート＞…作成するプログラムの論理部分を表したフローチャートです。ここで処理の流れを確認します。

＜関数定義＞…関数を複数使用する問題にのみ記載しています。また、関数同士の関係は「依存関係図」に示されます。それぞれの関数の役割や処理内容を指示しています。

フローチャート

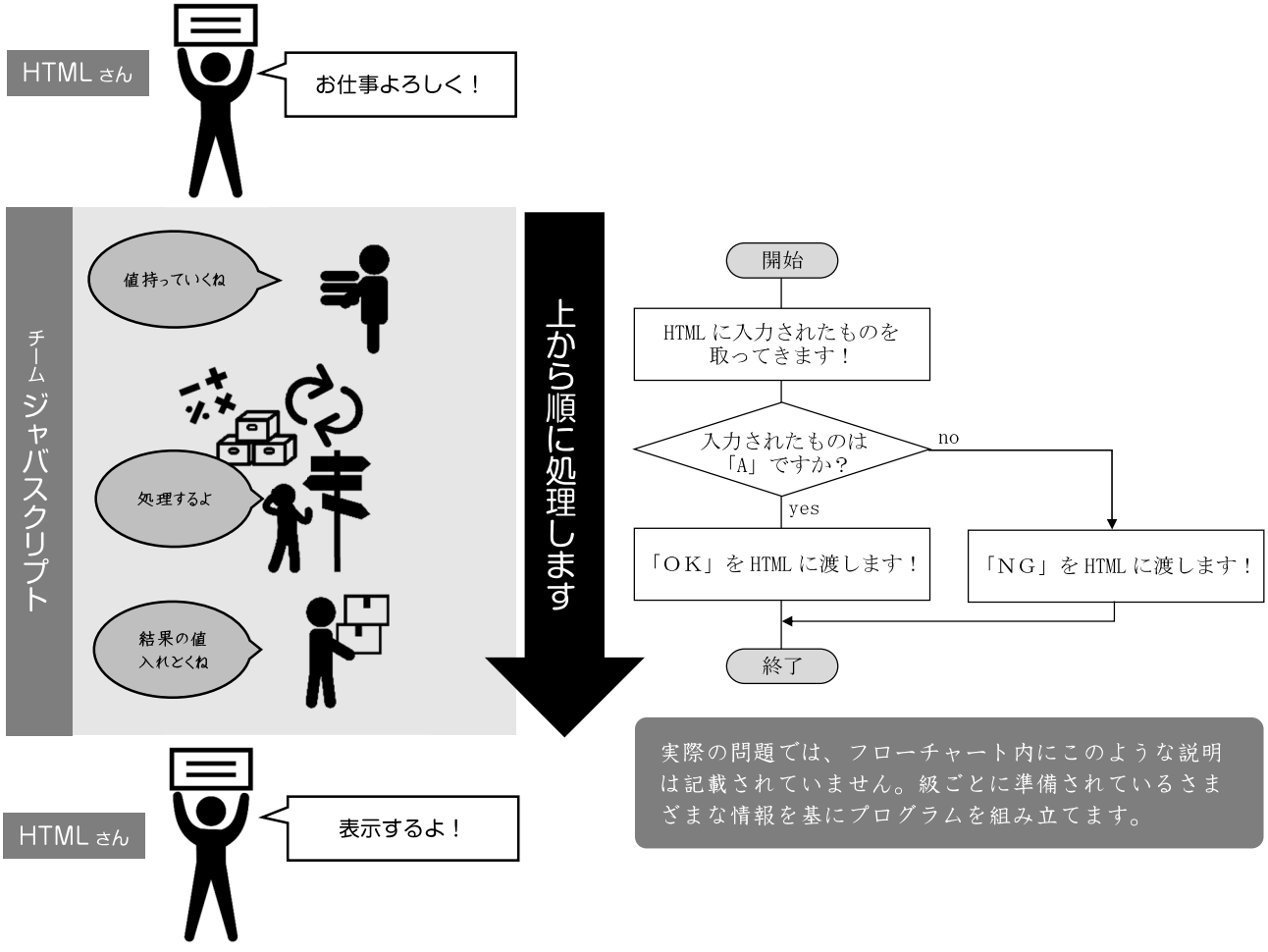
本試験で扱うフローチャートの記号は 6 種類です。

記　号	意　味
	プログラムの開始・終了
	条件による分岐
	処理
	繰り返しの始まり
	繰り返しの終わり
	制御の流れ

本試験問題では、入力する場所を設ける役割と結果を表示する役割は HTML が担い、
入力された値を取ってきたり、処理したり、結果を html に運んだりする役割を JavaScript が担います。

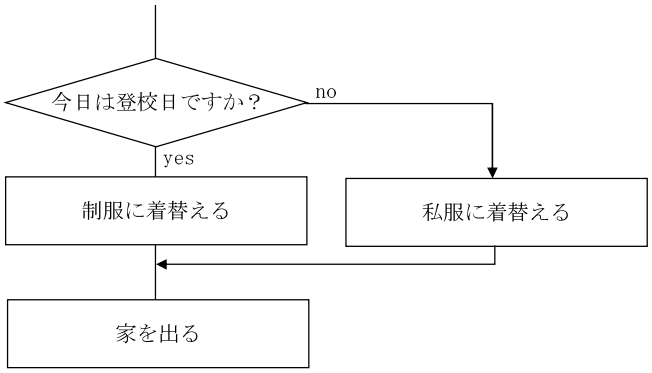
問題文に記載されているフローチャートは、この JavaScript の部分の流れを説明しています。

例）入力されたものが「A」だったら「ＯＫ」、違ったら「ＮＧ」と表示したい。



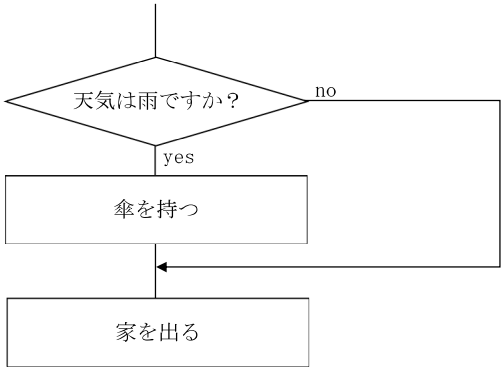
順次構造と分岐構造

yes か no かによって行うことが異なる場合




```
if (今日は登校日ですか？) {
    制服に着替える
}
else {
    私服に着替える
}
家を出る
```

yes のときにのみ行うことがあり、no のときは何もしない場合

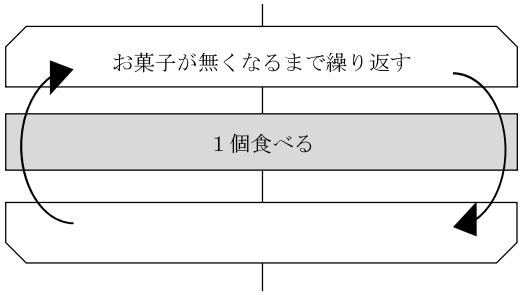



```
if (天気は雨ですか？) {
    傘を持つ
}
家を出る
```

反復構造（３級以上）

特定の条件が満たされるまで繰り返す（while）

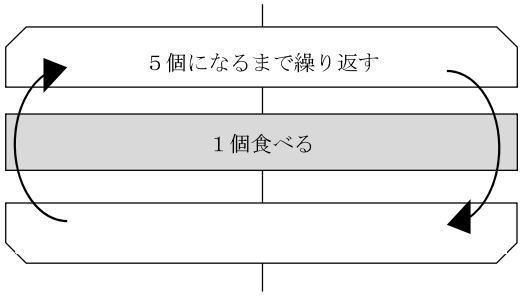
例）お菓子を「無くなるまで」食べる




```
while (お菓子が無くなるまで繰り返す) {
    １個食べる
}
```

特定の回数繰り返す（for）

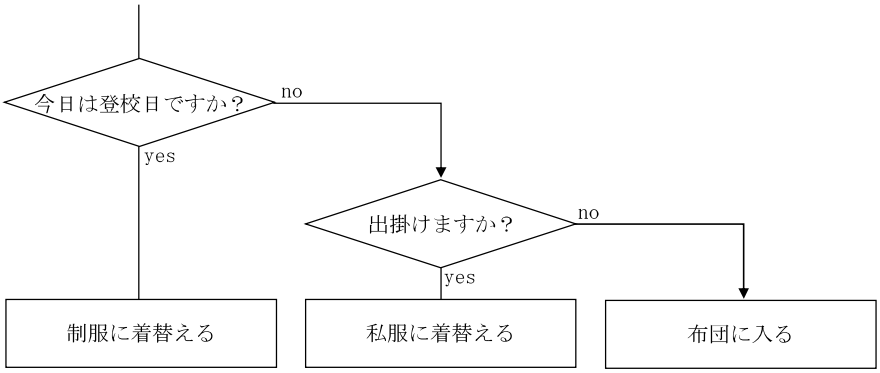
例）お菓子を「５個と決めて」食べる




```
for (5 個になるまで繰り返す) {
    １個食べる
}
```

条件分岐（else if）

yes か no か判定し、no だった場合にさらに yes か no か判定したい場合



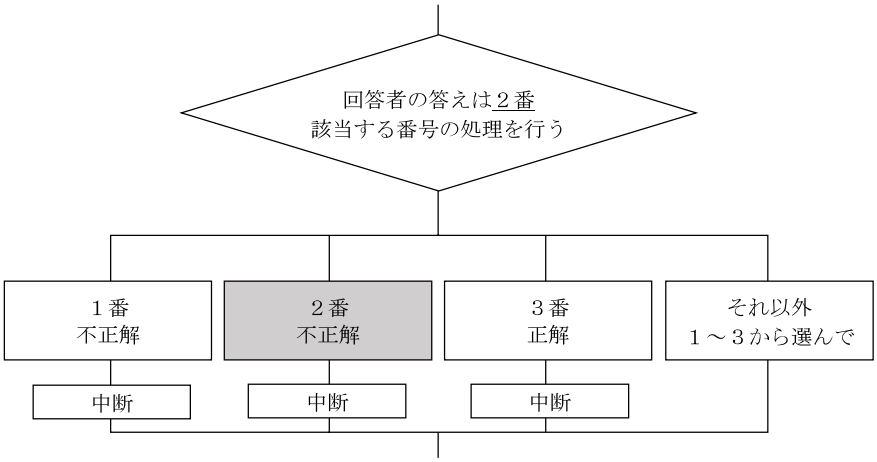

```
if (今日は登校日ですか？) {
   制服に着替える
}
else if (出掛けますか？){
   私服に着替える
}
else {
   布団に入る
}
```

条件分岐（switch）

複数の選択肢を持つ場合や特定の値に基づいて処理を分岐させる場合

例）「どんぶらこ」と流れてきたのは？ １～３番から選んで！　１番：みかん　２番：すいか　３番：もも

回答：２番　結果：不正解

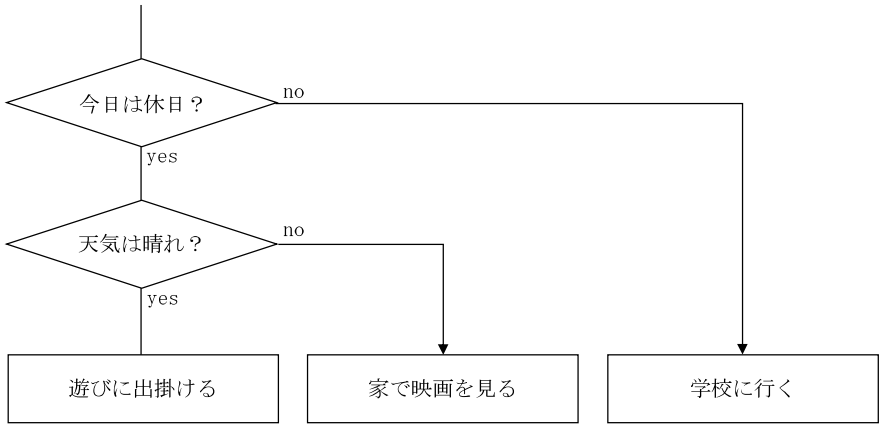



```
switch (回答：２番) {
   case １番
      不正解
      break
   case ２番
      不正解
      break
   case ３番
      正解
      break
   default
      １～３から選んで
}
```

switch では、一致するケースが見つかった後、明示的に中断（break）するまで、後続のケースの処理も通過して実行されます。そのため、中断（break）処理が必要となります。

条件分岐（if の入れ子）【１級】

複数条件で判定したい場合

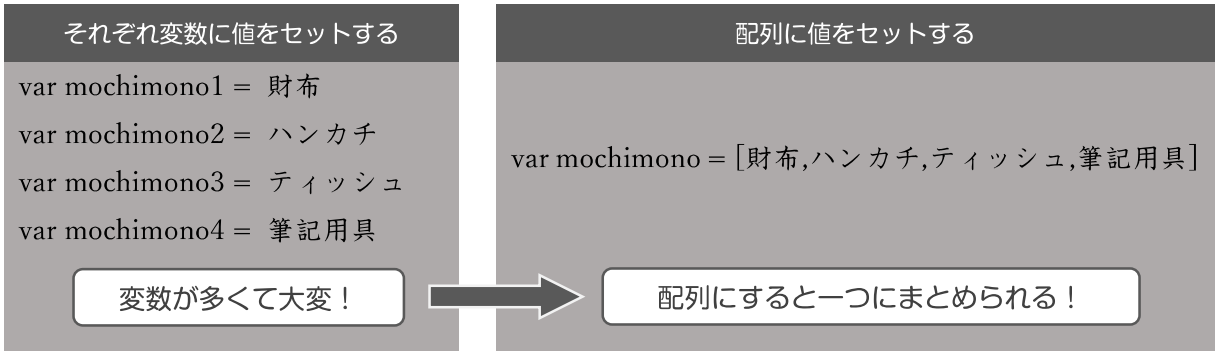



```
if (今日は休日？) {
   if (天気は晴れ？) {
      遊びに出掛ける
   }
   else {
      家で映画を見る
   }
}
else {
   学校に行く
}
```

配列【1級】

データの数が多いとき、それぞれに変数を準備することなく、一つの変数の中に複数のデータをまとめて管理することができる仕組みを「配列」といいます。
配列を作成するには、角カッコ［ ］を使用し、データはコンマ , で区切ります。

例）持ち物をリストアップする。



そして、そのセットされたデータを使う際には、そこからデータを指定する必要があります。そのときに使われるのが「添え字」（インデックス）です。
配列にセットされているデータには、最初のデータを「0」とし、順に 1、2…と番号が付きますが、この番号のことを「添え字」といいます。
変数の配列からデータを指定するときは、変数名[添え字]とします。

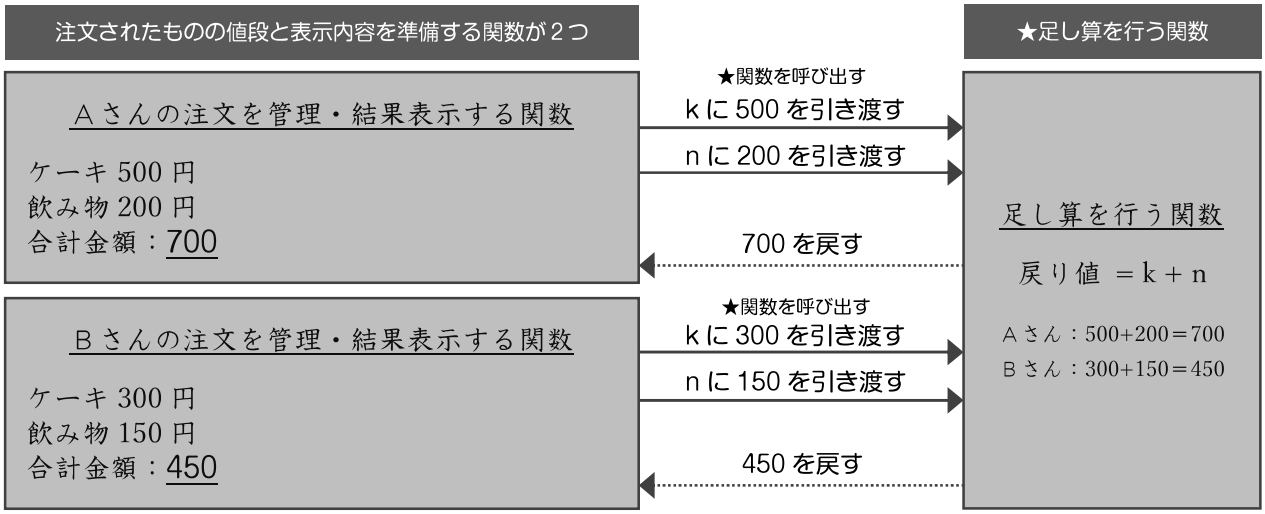
例）持ち物からハンカチを取り出したい。

```
var mochimono = [財布,ハンカチ,ティッシュ,筆記用具]
                  0     1       2         3
mochimono [1] ←1 番のハンカチを取り出す
```

複数の関数作成【1級】

本試験では、フォームのボタンが押されると関数（function）が動作し、そこに設定された一連の処理を行い、結果を導くつくりになっていますが、これはユーザ定義関数と呼ばれ、処理を部品化して再利用することができるメリットがあります。
1級では、さらにステップアップし、複数の関数を用意し、それらを組み合わせて処理をする出題があります。

例）足し算の処理のみ行う関数を準備し、呼び出して結果を出す。



JavaScript の基本ルール

全般

・プログラムは半角英数字・半角スペースで記述する。(文字列には全角使用可)

・大文字と小文字は別物として扱われる。(区別される)

・一つの文(命令)の終わりには「;」(セミコロン)を付ける。 ; は命令文の終わりを意味する。

※ブラウザが推測して解釈してくれるため、省略が可能。ただし、思わぬ動きをする可能性もあるため付けることが推奨される。

・文字列は「'」(シングルクォーテーション)または「"」(ダブルクォーテーション)で囲む。

他に「`」(バッククォート)で囲む方法もある。

主な演算子

種類	意味
=	代入
==	等しい
!=	等しくない
<	未満(〜より小さい)
<=	以下
>	超過(〜より大きい)
>=	以上

種類	意味
+	数値の足し算(加算)
	文字列の結合
−	引き算(減算)
*	かけ算(乗算)
/	割り算(除算)
++	1増やす
−−	1減らす

種類	意味
+=	加算代入
−=	減算代入
*=	乗算代入
/=	除算代入
,	式の区切り
.	〜の
!	ではない

変数の命名ルール

・数字のみ、先頭が数字の名前は禁止。(NG 例:3name　OK 例:name3)

・予約語と呼ばれる JavaScript で別の目的で既に使用することが決まっている名前は使用不可。

(NG 例:if、for、break など)

・大文字と小文字が厳密に区別される。(例:Box と box は別物となる)

型について

文字列や数値などのデータの種類のことを「型 (Type)」と呼ぶ。

■よく使われる型

型名	種類	例
Number	数値	0、123、-4、1.5
String	文字列	'こんにちは'、"あいうえお"
Boolean	論理値	true、false

■型の変換

＋演算子は数値の場合は足し算 (加算)、文字列の場合は結合になる。

文字列とその他の型を＋演算した場合、文字列に変換される。

例	結果	結果の型
1 + 2	3	Number 型(数値)
'1' + '2'	12	String 型(文字列)
1 + '2'	12	String 型(文字列)
'1' + 2	12	String 型(文字列)

他にも、あらかじめ関数を用いて型変換を行う方法もある。

本試験では、文字列を数値型に型変換する際、Number( ) ; 関数を使用。