令和5年度文部科学省委託「専修学校による地域産業中核的人材養成」事業

フロントエンドエンジニア
教材資料 2

IT 分野 DX 人材養成のモデルプログラム開発と実証事業

令和5年度文部科学省委託「専修学校による地域産業中核的人材養成」事業

フロントエンドエンジニア
教材資料 2

IT 分野 DX 人材養成のモデルプログラム開発と実証事業

目 次

フロントエンドエンジニア養成講座

JavaScript

1

はじめに

【学習目標】

本テキストでは、JavaScriptを理解することを目標とします。

目次

2

JavaScript
第１章
JavaScript基礎

【本章学習内容】

本章では、JavaScriptの基本文法について学習します。

3

第１章　JavaScript基礎

１.１　JavaScriptの記述方法

１.１.１　記述方法のメリット／デメリット

- HTMLファイルの<script>～</script>タグ内に記述
- HTMLファイルとは別の外部ファイル（JavaScriptファイル）に記述

	HTMLファイルに記述する場合	外部ファイルに記述する場合
ファイル拡張子	～.html	～.js
メリット	HTMLファイルにHTMLとJavaScriptのプログラムを併記するので、各プログラムの確認が容易である	JavaScriptプログラムを複数のHTMLファイルに反映できる
デメリット	JavaScriptプログラムは、それを記述したHTMLファイルにしか反映できない	HTMLファイルと外部ファイルをそれぞれ開いて各プログラムを確認しなければならない

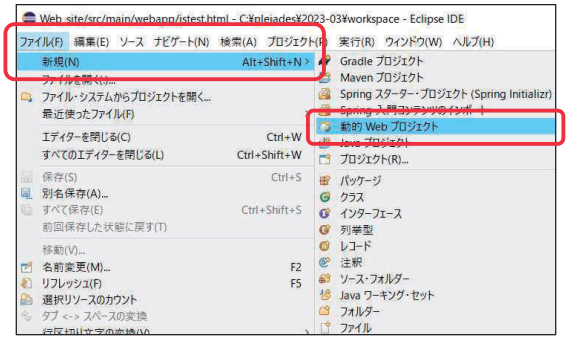
4

第１章　JavaScript基礎

１.１　JavaScriptの記述方法

１.１.２　HTMLファイルへの記述

- Eclipseを起動
- メニューの「ファイル」→「新規」→「動的Webプロジェクト」を選択



第１章　JavaScript基礎

１.１　JavaScriptの記述方法

１.１.２　HTMLファイルへの記述

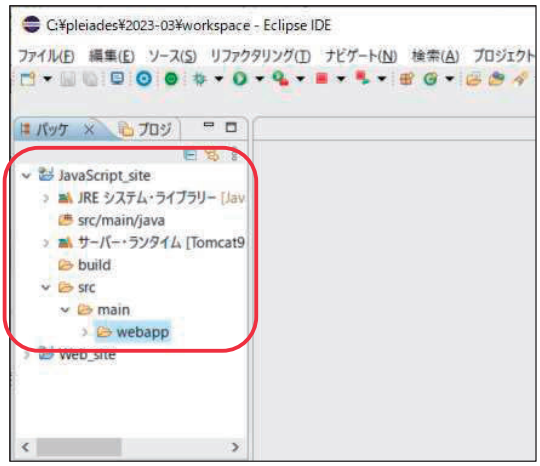
- プロジェクト名「JavaScript_site」
- プロジェクト名を入力後、「完了」ボタンをクリック

第１章　JavaScript基礎

１.１　JavaScriptの記述方法

１.１.２　HTMLファイルへの記述

・「JavaScript_site」→「src」→「main」→「webapp」の順番にクリックしフォルダを展開



7

第１章　JavaScript基礎

１.１　JavaScriptの記述方法

１.１.２　HTMLファイルへの記述

・メニューの「ファイル」→「新規」→「HTMLファイル」を選択

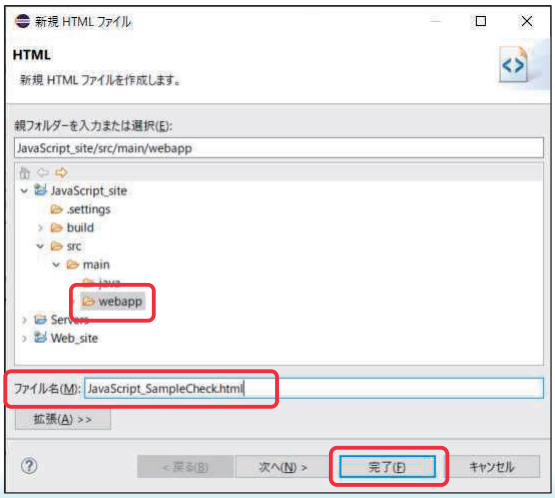
8

第１章　JavaScript基礎

１.１　JavaScriptの記述方法

１.１.２　HTMLファイルへの記述

- 保存フォルダ「webapp」
- ファイル名
  「JavaScript_SampleCheck.html」
- ファイル名を入力し「完了」ボタンをクリック



9

第１章　JavaScript基礎

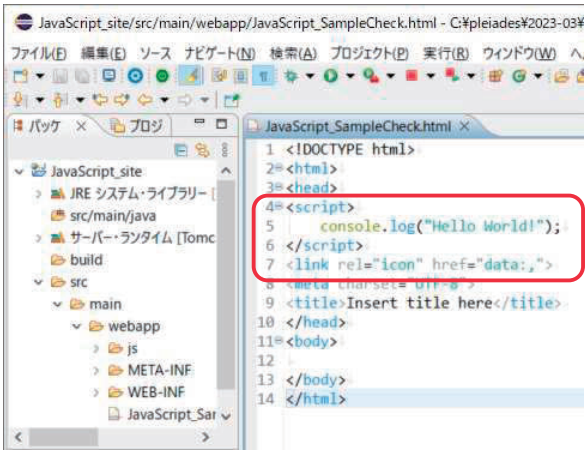
１.１　JavaScriptの記述方法

１.１.２　HTMLファイルへの記述

- HTMLファイル内にJavaScriptのプログラムを記述
- 下記コードの４～７行目を追加

```
   :
03  <head>
04  <script>
05     console.log("Hello World!");
06  </script>
07  <link rel="icon" href="data:,">
08  <meta charset="UTF-8">
   :
```

セミコロン忘れに注意



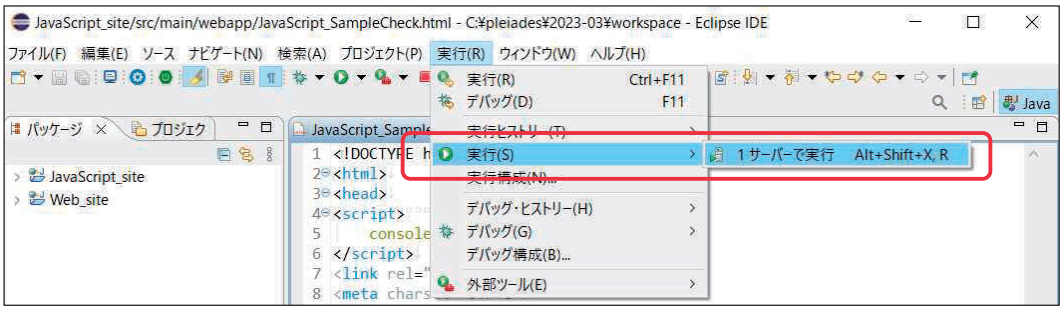
10

第１章　JavaScript基礎

１.１　JavaScriptの記述方法

１.１.２　HTMLファイルへの記述

・メニューの「実行」→「実行」→「サーバで実行」を選択

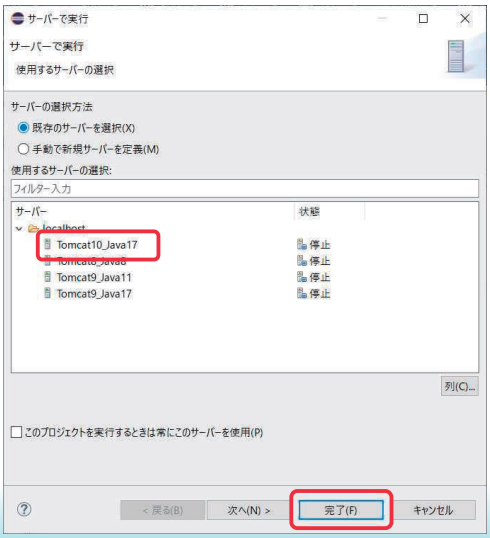


第１章　JavaScript基礎

１.１　JavaScriptの記述方法

１.１.２　HTMLファイルへの記述

・最新のWebサーバ（Tomcat）を選択して「完了」ボタンをクリック

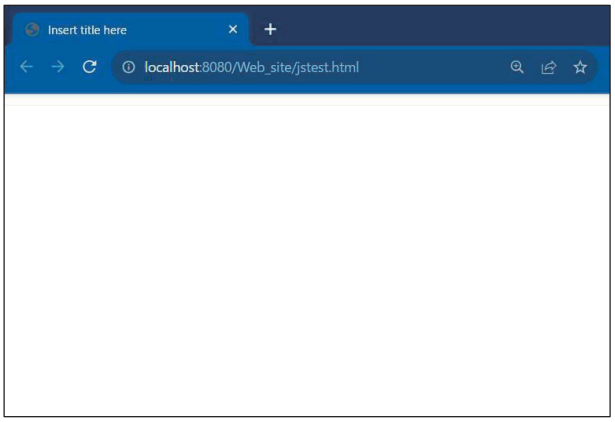


第１章　JavaScript基礎

１.１　JavaScriptの記述方法

１.１.２　HTMLファイルへの記述

- Webブラウザに空白ページが表示される
- JavaScriptのプログラムの実行結果は開発者ツール内に表示される
- WebブラウザのF12キーを押して「開発ツール」を起動



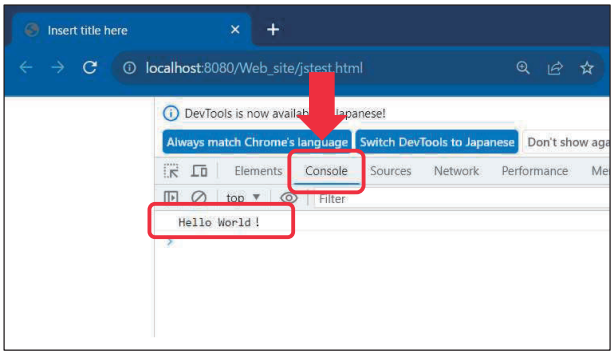
13

第１章　JavaScript基礎

１.１　JavaScriptの記述方法

１.１.２　HTMLファイルへの記述

- 開発者ツールの「Console」タブをクリックしてコンソール表示に切り換え
- JavaScriptの「console.log()」メソッドによって、コンソールに文字列「Hello World！」が出力



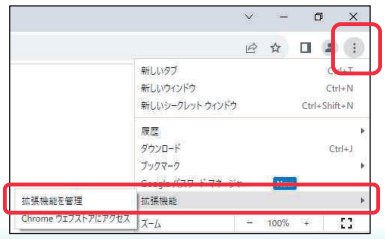
14

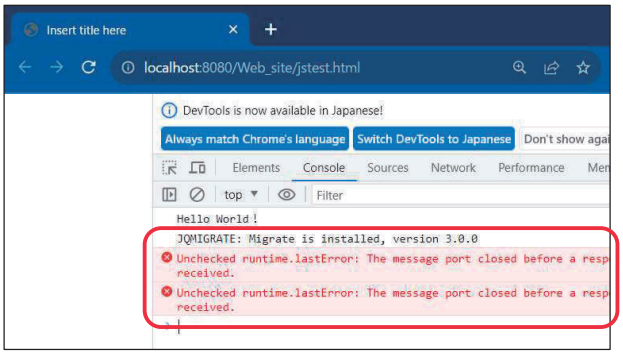
第１章　JavaScript基礎

１.１　JavaScriptの記述方法

１.１.２　HTMLファイルへの記述

- 「Unchecked runtime.lastError：…」と表示される現象は、Chromeのセキュリティ機能拡張が原因
- Chromeの機能拡張からセキュリティ機能拡張をOFFで対処





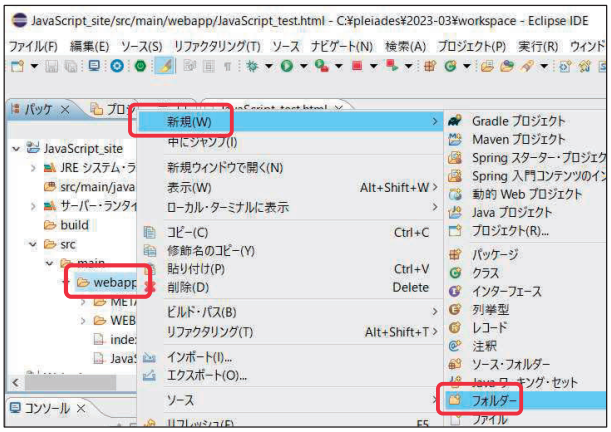
15

第１章　JavaScript基礎

１.１　JavaScriptの記述方法

１.１.３　外部ファイルへの記述

- 一般的に、外部ファイルとHTMLファイルは分けて保存
- 「webapp」フォルダの下に外部ファイル用の保存フォルダを作成
- 「webapp」フォルダを選択して、メニューの「ファイル」→「新規」→「フォルダー」を選択



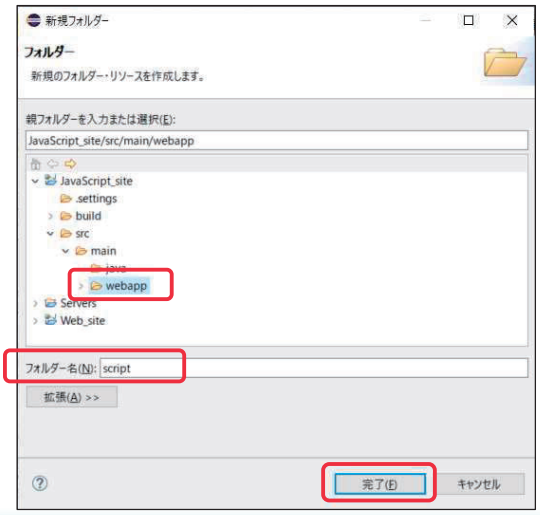
16

第１章　JavaScript基礎

１.１　JavaScriptの記述方法

１.１.３　外部ファイルへの記述

- 保存フォルダ「webapp」
- フォルダ名「script」
- ファイル名を入力し「完了」ボタンをクリック



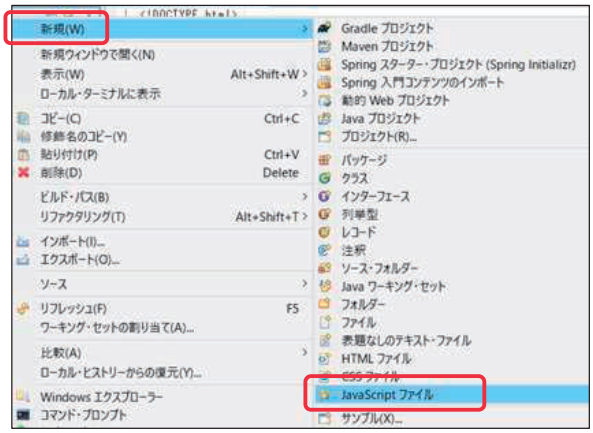
17

第１章　JavaScript基礎

１.１　JavaScriptの記述方法

１.１.３　外部ファイルへの記述

- 「script」フォルダを選択して、メニューの「ファイル」→「新規」→「 JavaScriptファイル」を選択



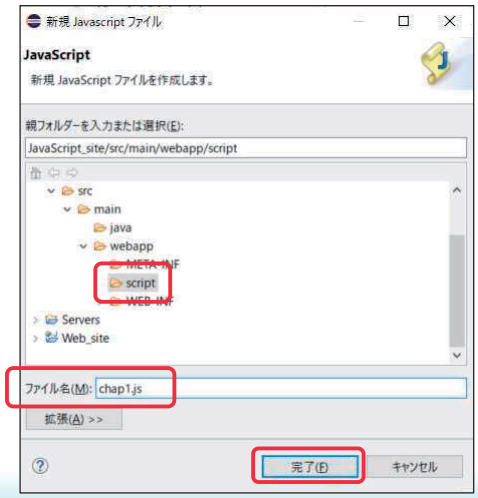
18

第１章　JavaScript基礎

１.１　JavaScriptの記述場所

１.１.３　外部ファイルへの記述

・保存フォルダ「script」
・ファイル名「chap1.js」
・ファイル名を入力し「完了」ボタンをクリック



19

第１章　JavaScript基礎

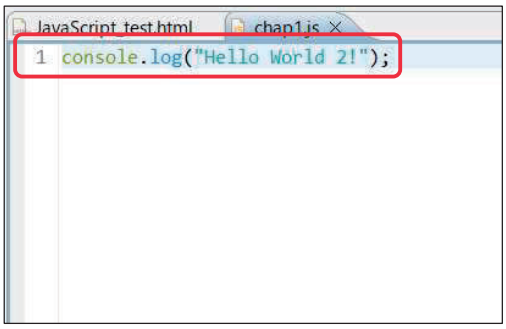
１.１　JavaScriptの記述方法

１.１.３　外部ファイルへの記述

・JavaScriptプログラムの雛形のコード
　/**
　　*
　　*/
　は不要なので消去する
・下記コードを追加

セミコロン忘れに注意

```
01  console.log("Hello World 2!");
```



20

第１章　JavaScript基礎

１.１　JavaScriptの記述方法

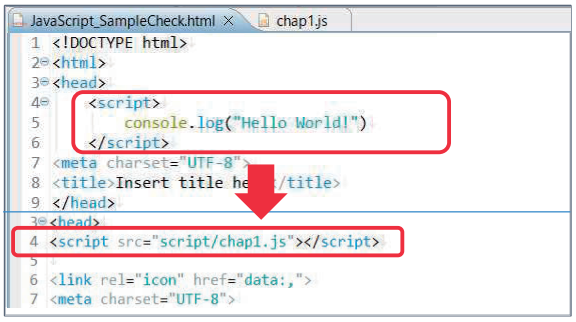
１.１.３　外部ファイルへの記述

・HTMLファイル
「JavaScript_sampleCheck.html」
の３～５行目を消去

```
04　<script>
05　　console.log("Hello World!");
06　</script>
```

・４行目に以下のコードを追加

```
04　<script src="script/chap1.js"></script>
```



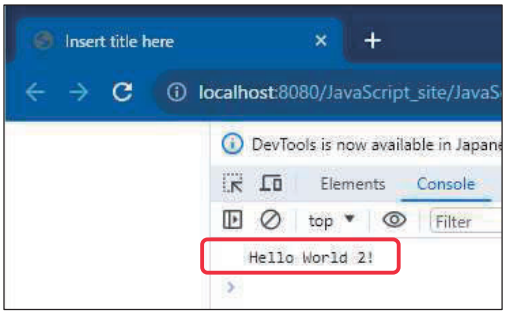
第１章　JavaScript基礎

１.１　JavaScriptの記述方法

１.１.３　外部ファイルへの記述

・Webブラウザの「再読み込みボタン」をクリック

・JavaScriptの「console.log()」メソッドによって、コンソールに文字列「Hello World 2!」が出力



第１章　JavaScript基礎

１.１　JavaScriptの記述方法

１.１.４　HTMLの<script>タグ

- WebページでJavaScriptなどのプログラムを実行するために<script>タグを使用
- HTMLファイルに記述したJavaScriptプログラムを実行する方法と外部ファイルに記述したJavaScriptプログラムを実行する方法がある
- 一般的にJavaScriptプログラムは外部ファイルに記述
- 外部ファイルに記述されたJavaScriptプログラムを読み込む場合、HTMLファイルに<script>タグを記述し、src属性にJavaScriptのファイルパスを相対パスで指定
- <script>タグのtype属性にスクリプト言語の種類を指定
- JavaScriptプログラムを記述する場合はtype属性の指定を省略
- <script>タグは<head>タグ内、<body>タグ内のどちらにも記述可能
- 記述位置によってJavaScriptプログラムの実行されるタイミングが異なる

第１章　JavaScript基礎

１.２　変数

１.２.１　変数とは

- 変数とは、プログラムで扱う値を代入したり、取り出したりするための入れ物
- 入力ミスを発見しやすくするため変数は事前に宣言・定義することが推奨される
- 変数に値のデータ型（種類）を気にせず代入可能
- 変数名の付け方のルール

１文字目は、半角英字、アンダースコア記号、ドル記号が使用できる
２文字目以降は、数字も使用できる
予約語と同じ名前を変数名に付けることはできない

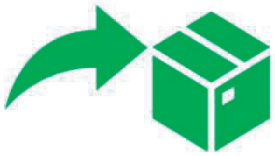
（良い例）
value　… 英単語を使用
userName　… 英単語の組み合わせ

（悪い例）
v　… 短すぎて推測が難しい
yuzamei　… ローマ字は理解できない

第１章　JavaScript基礎

１.２　変数

１.２.１　変数とは



・宣言（定義）できる変数の種類と特長

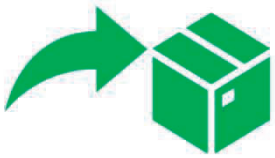
	再代入	再宣言
let変数	○	×
var変数	○	○
const変数	×	×

25

第１章　JavaScript基礎

１.２　変数

１.２.２　let変数



・プログラムの先頭に 'use strict' を記述
・変数宣言
【書式】let 変数名;
（例）　let value;
・変数への値の代入
【書式】変数名 = 値;
（例）　value = 100;

代入

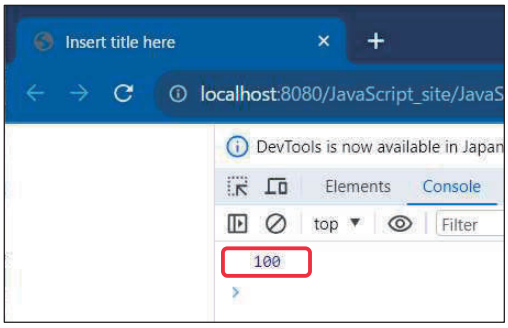
・変数は値を何度も代入できる（再代入できる）
・同じ関数内で同じ変数名を指定して宣言できない（再宣言できない）

26

第１章　JavaScript基礎

```
01  'use strict';
02
03  let value;
04  value = 100;
05  console.log(value);
```

・実行結果

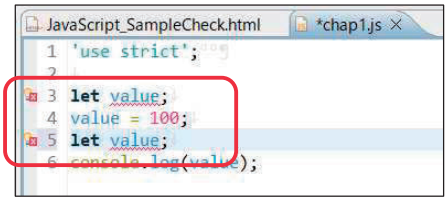


27

第１章　JavaScript基礎

```
01  'use strict';
02
03  let value;
04  value = 100;
05  let value;      ←この行を追加
06  console.log(value);
```

・実行結果



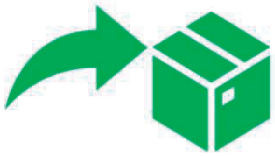
※3行目と5行目にエラーが表示される

28

第１章　JavaScript基礎

1.2　変数

1.2.3　var変数



・変数宣言

【書式】var 変数名;

（例）　var value2;

・変数への値の代入

【書式】変数名 = 値;

（例）　value2 = 200;

代入

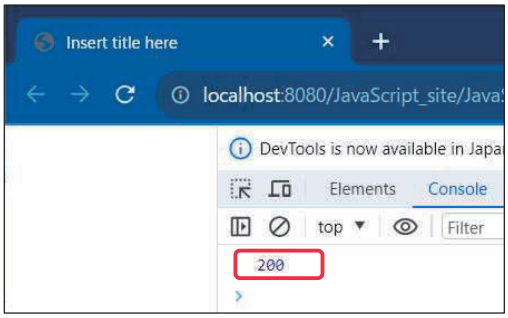
・変数は値を何度も代入できる（再代入できる）

・同じ関数内で同じ変数名を指定して宣言できる（再宣言できる）

第１章　JavaScript基礎

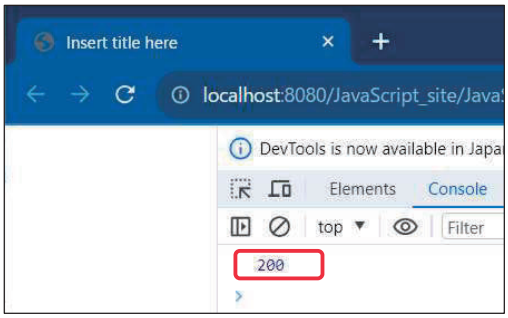
```
01  'use strict';
02
03  var value2;
04  value2 = 200;
05  console.log(value2);
```

・**実行結果**



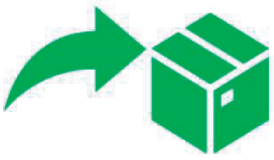
```
01  'use strict';
02
03  var value2;
04  value2 = 200;
05  var value2;      ←この行を追加
06  console.log(value2);
```

・実行結果



1.2　変数

1.2.4　const変数



・変数宣言と値の代入

【書式】const 変数名 = 値;

（例）　const value3 = 300;

変数宣言と同時に代入

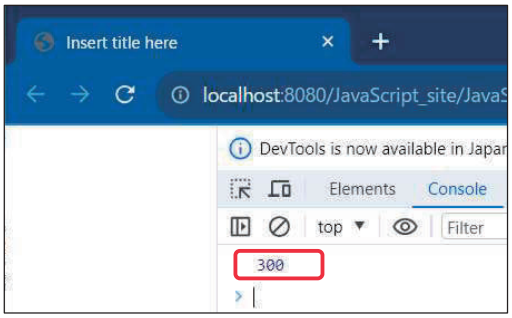
・変数は値を一度だけ代入できる（再代入できない）

・同じ関数内で同じ変数名を指定して宣言できない（再宣言できない）

第１章　JavaScript基礎

```
01  'use strict';
02
03  const value3 = 300;
04  console.log(value3);
```

・実行結果

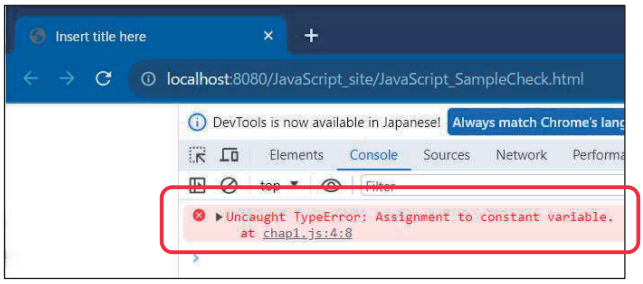


33

第１章　JavaScript基礎

```
01  'use strict';
02
03  const value3 = 300;
04  value3 = 400;
05  console.log(value3);
```

・実行結果

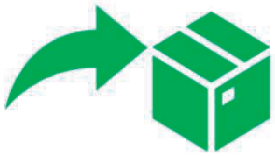


34

第１章　JavaScript基礎

1.2　変数

1.2.5　コメント



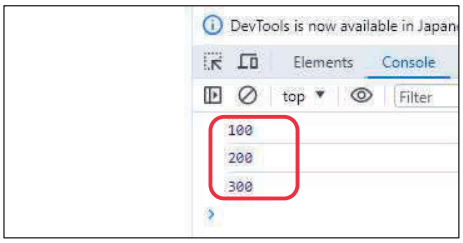
- プログラムに注釈（実行されない部分）を記述
- 「//」から行末までの注釈は単一行コメント
- 「/*」と「*/」で囲まれた注釈は複数行コメント
- 注釈の書き方は、開発プロジェクトのコーディング規約に従う
- プログラムにコメントを適切に入れることで、プログラムの保守性が向上
- 入力した命令を一時的に無効化したい場合にコメントを活用

35

第１章　JavaScript基礎

```
01  'use strict';
02
03  /* let変数の使用例 */
04  let value;
05  value = 100;
06  //let value;   エラーになる
07  console.log(value);
08
09  /* var変数の使用例 */
10  var value2;
11  value2 = 200;
12  var value2;      //エラーにならない
13  console.log(value2);
14
15  /* const変数（定数）の使用例 */
16  const value3 = 300;
17  //value3 = 400;   エラーになる
18  console.log(value3);
```

・実行結果



36

1.3　演算子

1.3.1　代入演算子と算術演算子

・変数への代入や計算で使用

記号	意味
=	代入
+	加算（足し算）
-	減算（引き算）
*	乗算（掛け算）
/	除算（割り算）
%	剰余算（割り算の余りを求める）
++	インクリメント（値を１増やす）
--	デクリメント（値を１減らす）


```
01  'use strict';
02
03  let x = 500, y = 200;    //代入演算子
04  let ans;
05  ans = x + y;             //加算（足し算）
06  console.log(ans);
07  ans = x - y;             //減算（引き算）
08  console.log(ans);
09  ans = x * y;             //乗算（掛け算）
10  console.log(ans);
11  ans = x / y;             //除算（割り算）
12  console.log(ans);
13  ans = x % y;             //剰余残（割り算の余りを求める）
14  console.log(ans);
```

・**実行結果**

```
700
300
100000
2.5
100
```

第１章　JavaScript基礎

１.３　演算子

１.３.１　代入演算子と算術演算子

・インクリメント演算子は、変数の値を１増やす
　【書式】変数名++　または　++変数名
　（例）　i++;　… 後置式
　　　　++i;　… 前置式
・デクリメント演算子は、変数の値を１減らす
　【書式】変数名--　または　--変数名
　（例）　j++;　… 後置式
　　　　++j;　… 前置式
・変数はインクリメントとデクリメントの演算結果で自動更新

第１章　JavaScript基礎

```
01  'use strict';
02
03  let i = 1;
04  console.log(i);
05  i++;          //後置式のインクリメント
06  console.log(i);
07  ++i;          //前置式のインクリメント
08  console.log(i);
09  let j = 10;
10  console.log(j);
11  j--;          //後置式のデクリメント
12  console.log(j);
13  --j;          //前置式のデクリメント
14  console.log(j);
```

・**実行結果**

```
1
2
3
10
9
8
```

※注意
「i = i++;」や「j = --j;」と記述しない

1.3　演算子

1.3.2　複合代入演算子

・変数への代入と計算を同時に指定

記号	使用例	備考
+=	a += 10;	a = a + 10; と同じ
-=	a -= 10;	a = a - 10; と同じ
*=	a *= 10;	a = a * 10; と同じ
/=	a /= 10;	a = a / 10; と同じ
%=	a %= 10;	a = a % 10; と同じ

【書式】変数 += 値;　または　変数 += 計算式;

（例）　a += 10;

　　b -= (c + d);


```
01  'use strict';
02
03  let a = 0;
04  a += 10;          //足し算の複合代入演算子
05  console.log(a);
06
07  let b = 100;
08  let c = 20, d = 10;
09  b -= (c + d);     //引き算の複合代入演算子
10  console.log(b);
```

・**実行結果**

```
10
70
```

第１章　JavaScript基礎

１.４　配列

１.４.１　配列とは

- 配列とは、プログラムで扱う値を代入したり、取り出したりするための、連続して並んでいる入れ物
- 各入れ物が一つの方向に並ぶ配列を一次元配列という
- 配列名は、入れ物全体を示す
- 添字（インデックス）は、各入れ物に付けられた、０から始まる連続番号
- 要素は、配列名と添字の組み合わせで指定される各入れ物
- JavaScriptでは、配列は事前に宣言・定義することが推奨される
- JavaScriptでは、配列の要素数はプログラム実行中に調整可能

（例）pref[0]　… この要素に値を代入できる

pref[5]　… 右上図ではpref[5]は存在しないが、この要素に値を代入できる

43

第１章　JavaScript基礎

１.４　配列

１.４.１　配列とは

- 配列宣言と初期化

【書式①】const 配列名 = [];

（例）
```
const pref1 = [];
pref1[0] = "東京";
pref1[1] = "埼玉";
pref1[2] = "千葉";
pref1[3] = "神奈川";
pref1[4] = "茨城";
```

【書式②】
```
const 配列名 = [値, 値, 値, 値, 値, …];
const pref2 = ["東京", "埼玉", "千葉", "神奈川", "茨城"];
```

44

```
01  'use strict';
02
03  //書式①
04  const pref1 = [ ];
05  pref1[0] = "東京";
06  pref1[1] = "埼玉";
07  pref1[2] = "千葉";
08  pref1[3] = "神奈川";
09  pref1[4] = "茨城";
10  console.log(pref1);
11
12  //書式②
13  const pref2 = ["東京", "埼玉", "千葉", "神奈川", "茨城"];
14  console.log(pref2);
```

・実行結果

```
Array(5)
  0: "東京"
  1: "埼玉"
  2: "千葉"
  3: "神奈川"
  4: "茨城"
Array(5)
  0: "東京"
  1: "埼玉"
  2: "千葉"
  3: "神奈川"
  4: "茨城"
```


1.4　配列

1.4.1　配列とは

・配列宣言と初期化

【書式③】const 配列名 = new Array();

（例）　const pref3 = new Array();
pref3[0] = "東京";
pref3[1] = "埼玉";
pref3[2] = "千葉";
pref3[3] = "神奈川";
pref3[4] = "茨城";

【書式④】const 配列名 = new Array(値, 値, 値, 値, 値, …);
const pref4 = new Array("東京", "埼玉", "千葉", "神奈川", "茨城");

```
01  'use strict';
02
03  //書式③
04  const pref3 = new Array( );
05  pref3[0] = "東京";
06  pref3[1] = "埼玉";
07  pref3[2] = "千葉";
08  pref3[3] = "神奈川";
09  pref3[4] = "茨城";
10  console.log(pref3);
11
12  //書式④
13  const pref4 = new Array("東京", "埼玉", "千葉", "神奈川", "茨城");
14  console.log(pref4);
```

・実行結果

```
Array(5)
  0: "東京"
  1: "埼玉"
  2: "千葉"
  3: "神奈川"
  4: "茨城"
Array(5)
  0: "東京"
  1: "埼玉"
  2: "千葉"
  3: "神奈川"
  4: "茨城"
```


1.4　配列

1.4.1　配列とは

- 【書式④】の「new Array(…)」において、小カッコ内に数値を一つだけ指定した場合は、特に注意が必要

  （例）const pref5 = new Array(2, 4, 6, 8, 10);

  → 要素数５の配列が用意され、初期値 {2, 4, 6, 8, 10} が代入される

  const pref6 = new Array(5);

  → 要素数５の配列が用意され、初期値は代入されない

- 配列宣言は予約語constの指定が望ましい

  （例）const pref7 = [2, 4, 6, 8, 10];　→ 初期値は代入できる

  pref7[0] = 12;　→ 要素は変更できる

  pref7 = ["東京", "埼玉", "千葉", "神奈川", "茨城"];　→ 文法エラーになる

  pref7 = new Array(5);　→ 文法エラーになる

```
01  'use strict';
02
03  const pref5 = new Array(2, 4, 6, 8, 10);
04  console.log(pref5);
05  const pref6 = new Array(5);
06  console.log(pref6);
07
08  const pref7 = [2, 4, 6, 8, 10];
09  pref7[0] = 12;
10  console.log(pref7);
11  //pref7 = ["東京", "埼玉", "千葉", "神奈川", "茨城"];    文法エラー
12  //pref7 = new Array(5);                                 文法エラー
```

・実行結果

```
Array(5)
  0: 2
  1: 4
  2: 6
  3: 8
  4: 10
Array(5)
  ※empty
Array(5)
  0: 12
  1: 4
  2: 6
  3: 8
  4: 10
```


１.４　配列

１.４.１　配列とは

- 配列は要素数を気にせず使用可能
  （例）const pref8 = new Array(5);　→ 要素数５の配列が用意される
  　　pref8[46] = "沖縄";　→ 要素に値を代入できる
- 値が代入されていない要素にはundefined値が自動設定
  （例）console.log(pref8[0]);　→ undefined値が出力される
- JavaScriptでは添字に負数を指定可能
  （例）pref8[-1] = "全国";　→ 要素に値を代入できる
  このような使い方は混乱の原因になるので避けた方がよい
- 配列の要素数の取得
  （例）console.log(pref8.length);　→ 要素数47が出力される
  （添字0以上の要素がカウント対象）

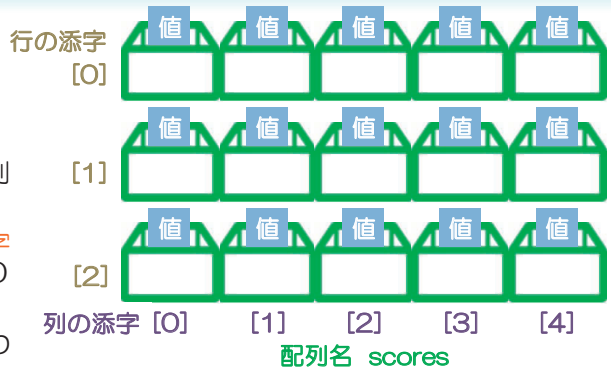
```
01  'use strict';
02
03  const pref8 = new Array(5);
04  pref8[46] = "沖縄";
05  console.log(pref8[46]);
06
07  console.log(pref8[0]);
08
09  pref8[-1] = "全国";
10  console.log(pref8[-1]);
11
12  console.log(pref8.length);
```

・実行結果

```
沖縄
undefined
全国
47
```


１.４　配列

１.４.２　二次元配列



- 配列の各入れ物が二つの方向に並ぶ配列を二次元配列という
- 行の添字（行インデックス）と列の添字（列インデックス）があり、それぞれ０から始まる連続番号
- 配列名、行の添字、列の添字を組み合わせて要素を指定
- JavaScriptでは、配列は事前に宣言・定義することが推奨される
- JavaScriptでは、配列の要素数はプログラム実行中に調整可能

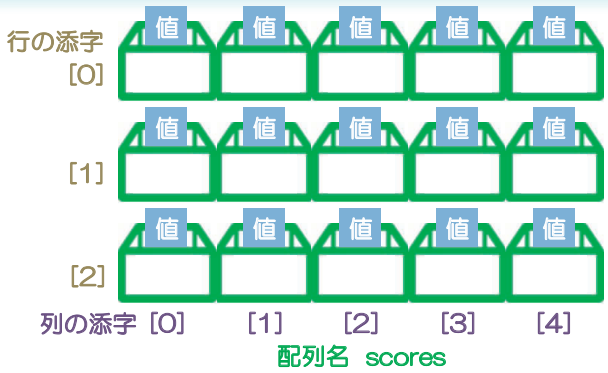
（例）scores[0][0]　… この要素に値を代入できる

scores[3][5]　… 右上図ではscores[3][5]は存在しないが、この要素に値を代入できる

第１章　JavaScript基礎

１.４　配列

１.４.２　二次元配列



- JavaScriptの仕様では、二次元配列を含む多次元配列は定義されていない
- 一次元配列の各要素に別の一次元配列を格納し、擬似的な二次元配列として扱う
- 配列宣言と初期化

【書式】const 配列名 = [[ ], [ ], [ ], …];

const 配列名 = [[値, 値, 値, …], [値, 値, 値, …], [値, 値, 値, …], …];

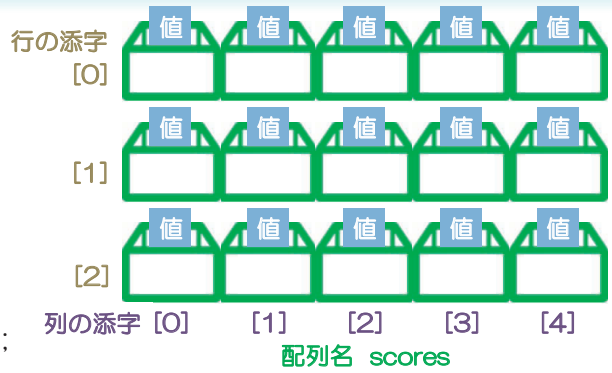
（例） const scores = [[67, 70, 85, 54, 98],
[83, 65, 46, 76, 88],
[75, 82, 91, 69, 77]];

53

第１章　JavaScript基礎

１.４　配列

１.４.２　二次元配列



- 行数（行方向の要素数）の取得

（例）console.log(scores.length);

→ 行数3が出力される

- 列数（列方向の要素数）の取得

（列）console.log(scores[0].length);

console.log(scores[1].length);

console.log(scores[2].length);

→ それぞれ列数5が出力される

- 一次元配列の利用時の注意事項は、二次元配列にも適用される

54

```
01  'use strict';
02
03  const scores = [ [67, 70, 85, 54, 98],
04                   [83, 65, 46, 76, 88],
05                   [75, 82, 91, 69, 77]  ];
06  console.log(scores[0][0]);
07  console.log(scores[1]);
08
09  console.log(scores.length);
10
11  console.log(scores[0].length);
12  console.log(scores[1].length);
13  console.log(scores[2].length);
```

・**実行結果**

```
67
Array(5)
   0: 83
   1: 65
   2: 46
   3: 76
   4: 88
3
5
5
5
```


１.４　配列

１.４.３　連想配列

・キー（文字列）と値をペアで扱う

・配列宣言

【書式】const 配列名 = {};

const 配列名 = {キー：値, キー：値, キー：値, …}

※キーは、ダブルクォート記号で囲まなくてもよい

（例）　const seasons = {spring：″春″,

summer：″夏″,

autumn：″夏″,

winter：″冬″};

１.４　配列

１.４.３　連想配列

・配列名とキーの組み合わせで要素を指定　　※添字（インデックス）は使用不可

（例）　console.log(seasons.spring);　　※ドット記法（配列名.キー）

console(seasons["autumn"]);　　※ブラケット記法（配列名["キー"]）

・キーと値のペアを後から末尾に追加可能

（例）　const seasons2 = {spring : "春", summer : "夏"};

seasons2.autumn = "秋";

seasons2["winter"] = "冬";

console.log(seasons2);　　→ 春・夏・秋・冬の順に表示される

※Webブラウザがchromeの場合、コンソールの表示内容を展開すると
キーと値のペアがキーの昇順（文字コードの昇順）に表示される


```
01  'use strict';
02
03  const seasons = {spring : "春",
04                   summer : "夏",
05                   autumn : "夏",
06                   winter : "冬"};
07
08  console.log(seasons.spring);
09  console.log(seasons["autumn"]);
10
11  const seasons2 = {spring : "春", summer : "夏"};
12  seasons2.autumn = "秋";
13  seasons2["winter"] = "冬";
14  console.log(seasons2);
```

・実行結果

```
春
秋
Object
    autumn: '秋'
    spring: '春'
    summer: '夏'
    winter: '冬'
```

第１章　JavaScript基礎

１.５　分岐構文

１.５.１　if-else文

・if-else文は、分岐条件により実行する処理を分岐できる

【書式】
```
//ifブロック
if( 分岐条件 ) {
   分岐条件が成立した場合の処理；
}
//elseブロック
else {
   分岐条件が成立しなかった場合の処理；
}
```

・分岐条件が成立した（true）場合はifブロックの処理を実行、
成立しなかった（false）場合はelseブロックの処理を実行

・プログラムの可読性を高めるため、各ブロックの { } は省略しない

・elseブロックは省略可能

59

第１章　JavaScript基礎

１.５　分岐構文

１.５.１　if-else文

・比較演算子

==	等価	左辺と右辺が同じ値ならtrue（データ型が異なってもよい）
!=	不等価	左辺と右辺が異なる値ならtrue（データ型が異なってもよい）
>	より大きい	左辺が右辺より大きければtrue
>=	以上	左辺が右辺と等しい、または右辺より大きければtrue
<	未満	左辺が右辺より小さければtrue
<=	以下	左辺が右辺と等しい、または右辺より小さければtrue
===	厳密等価	左辺と右辺が同じデータ型かつ同じ値ならtrue
!==	厳密不等価	左辺と右辺が異なるデータ型または異なる値ならtrue

※データ型：数値型、長整数型、文字列型、論理値型、undefined型、null型など

60

```
01  'use strict';
02  let data1 = 10;
03  let data2 = "10";
04  //同じ値
05  if(data1 == data2) {
06    console.log("data1とdata2は同じです。");
07  } else {
08    console.log("data1とdata2は違います。");
09  }
10  //同じデータ型かつ同じ値
11  if(data1 === data2) {
12    console.log("data1とdata2は厳密に同じです。");
13  } else {
14    console.log("data1とdata2は厳密には違います。");
15  }
```

・実行結果
data1とdata2は同じです。
data1とdata2は厳密には違います。


```
01  'use strict';
02  let data3 = 10;
03  //以上
04  if(data3 >= 10) {
05    console.log("data3は10以上です。")
06  } else {
07    console.log("data3は10以上ではありません。")
08  }
09  //より大きい
10  if(data3 > 10) {
11    console.log("data3は10より大きいです。")
12  } else {
13    console.log("data3は10より大きくありません。")
14  }
```

・実行結果
data3は10以上です。
data3は10より大きくありません。

1.5　分岐構文

1.5.1　if-else文

・論理演算子

&&	論理積	条件式1 && 条件式2 (例) 降水確率 <= 30 && 所持金 >= 10000	条件式1と条件式2の両方がtrueならtrue
||	論理和	条件式1 || 条件式2 (例) 降水確率 <= 30 || 所持金 >= 10000	条件式1と条件式2の一方がtrueならtrue
!	否定	!条件式 (例) !(所持金 >= 10000)	条件式のtrueとfalseを反転

※優先順位：！（高）、&&（中）、||（低）


```
01  'use strict';
02  let weather = 10, money = 12000;
03  //論理積
04  if(weather <= 30 && money >= 10000) {
05    console.log("天気が良い、かつ、お金がある。");
06  } else {
07    console.log("天気が悪い、または、お金がない。");
08  }
09  //論理和
10  weather = 40, money = 15000;
11  if(weather <= 30 || money >= 10000) {
12    console.log("天気が良い、または、お金がある。");
13  } else {
14    console.log("天気が悪い、かつ、お金がない。");
15  }
```

・実行結果

```
天気が良い、かつ、お金がある。
天気が良い、または、お金がある。
```

1.5　分岐構文

1.5.1　if-else文

・if-else文のifブロックまたはelseブロックに、別のif-else文を入れる構造を
ネスト（入れ子構造）という

（例）

```
//外側のif-else文
if(降水確率 >= 30) {
    //内側のif-else文①
    if(強風 == true) {
        console.log(〝タクシーで移動する。〝);
    } else {
        console.log(〝電車とバスで移動する。〝);
    }
} else {
    //内側のif-else文②
}
```



```
01  'use strict';
02  let weather2 = 50, windy = true;
03  if(weather2 >= 30) {
04      if(windy == true) {
05          console.log("タクシーで移動する。");
06      } else {
07          console.log("電車とバスで移動する。");
08      }
09  } else {
10      if(windy == true) {
11          console.log("電車とバスで移動する。");
12      } else {
13          console.log("自転車で移動する。");
14      }
15  }
```

・実行結果
タクシーで移動する。

1.5　分岐構文

1.5.2　if-else if-else文

・if-else if-else文は、複数の分岐条件により実行する処理を細かく分岐できる

【書式】
```
//ifブロック
if( 分岐条件① ) {
   分岐条件①が成立した場合の処理；
}
//else-ifブロック
else if( 分岐条件② ) {
   分岐条件②が成立した場合の処理；
}
//elseブロック
else {
   分岐条件①と分岐条件②が両方とも成立しなかった場合の処理；
}
```

・else ifブロックは複数記述可能、elseブロックは省略可能


```
01  'use strict';
02  let weather3 = 30;
03  if(weather3 >= 40) {
04     //降水確率が40%以上の場合
05     console.log("長傘を持って出掛ける。");
06  } else if(weather3 >= 20) {
07     //降水確率が20%以上の場合
08     console.log("折りたたみ傘を持って出掛ける。");
09  } else {
10     //降水確率がそれ以外（20%未満）の場合
11     console.log("傘を持たずに出掛ける。");
12  }
13  console.log("出発！");
```

・実行結果
折りたたみ傘を持って出掛ける。
出発！

第１章　JavaScript基礎

１.５　分岐構文

１.５.３　switch文

・switch文は、複数の定数により実行する処理を細かく分岐できる

【書式】switch(変数または計算式) {

```
case 定数１：
   変数または計算式の結果が定数１と一致する場合の処理；
   break；     //switch文の実行を終了する
case 定数２：
   変数または計算式の結果が定数２と一致する場合の処理；
   break；     //switch文の実行を終了する
default：
   変数または計算式の結果がすべての定数と一致しない場合の処理；
}
```

69

第１章　JavaScript基礎

１.５　分岐構文

１.５.３　switch文

・case節は複数記述可能

```
//case節
case 定数：
   変数または計算式の結果が定数と一致する場合の処理；
   break；
```

・上のcase節から順に、変数または計算式が定数と一致するか判断される

・break文を記述すると、switch文の実行が終了

・break文は省略できるが、その場合は直後のcase節が続けて実行される

・case節の終わりにdefault節を記述、default節は省略可能

```
//default節
default：
   変数または計算式の結果がすべての定数と一致しない場合の処理；
```

70

```
01  'use strict';
02  const city = "大阪";
03  switch(city) {
04  case "東京":
05    console.log("関東地方です。");
06    break;
07  case "大阪":
08    console.log("関西地方です。");
09    break;
10  case "愛知":
11    console.log("中部地方です。");
12    break;
13  default:
14    console.log("その他の地方です。");
15  }
```

・実行結果
関西地方です。


```
01  'use strict';
02  const month = 6;
03  switch(month) {
04  case 3:
05  case 4:
06  case 5:
07    console.log("季節は春です。");
08    break;
09  case 6:
10  case 7:
11  case 8:
12    console.log("季節は夏です。");
13    break;
14  case 9:
15  case 10:
16  case 11:
17    console.log("季節は秋です。");
18    break;
19  case 12:
20  case 1:
21  case 2:
22    console.log("季節は冬です。");
23    break;
24  default:
25    console.log("季節は不明です。");
26  }
```

・実行結果
季節は夏です。

１.６　繰り返し構文

１.６.１　while文

・while文は、反復条件（継続条件）により実行する処理を繰り返す

【書式】while(反復条件) {
　　反復条件が成立した場合の処理；
　}

・wile文の動作

①反復条件を判定し、成立した（true）場合は②に進む
　成立しなかった（false）場合はwhile文の実行を終了する

②whileブロックの処理を１回実行して①に戻る

・プログラムの可読性を高めるため、whileブロックの { } は省略しない


```
01  'use strict';
02  console.log("while文①");
03  let i = 0;              //変数ｉに０を代入する
04  while(i < 5) {          //変数ｉが５未満の間、繰り返す
05    console.log((i + 1) + "周目");
06    i++;                  //変数ｉを１増やす
07  }
08  console.log("while文②");
09  let j = 1;              //変数ｊに１を代入する
10  while(j <= 5) {         //変数ｊが５以下の間、繰り返す
11    console.log(j + "周目");
12    j++;                  //変数ｊを１増やす
13  }
```

・実行結果

```
while文①
1周目
2周目
3周目
4周目
5周目
while文②
1周目
2周目
3周目
4周目
5周目
```

１.６　繰り返し構文

１.６.２　for文

・for文は、カウンタ（変数）の制御により実行する処理を繰り返す

【書式】for(カウンタの初期化；反復条件；カウンタの更新) {
　　反復条件が成立した場合の処理；
　}

・for文の動作

①カウンタに初期値を代入して②に進む

②反復条件を判定し、成立した（true）場合は③に進む
　成立しなかった（false）場合はfor文の実行を終了する

③forブロックの処理を１回実行して④に進む

④カウンタを更新して②に戻る

・プログラムの可読性を高めるため、forブロックの { } は省略しない


```
01  'use strict';
02  console.log("for文①");
03  for(let s = 0; s < 5; s++) {     //カウンタｓは０～４で動作する
04    console.log((s + 1) + "周目");
05  }
06  console.log("for文②");
07  for(let t = 1; t <= 5; t++) {    //カウンタｔは１～５で動作する
08    console.log(t + "周目");
09  }
```

・実行結果

```
for文①
1周目
2周目
3周目
4周目
5周目
for文②
1周目
2周目
3周目
4周目
5周目
```

１.６　繰り返し構文

１.６.３　for-of文

・for-of文は、配列の要素により実行する処理を繰り返す

【書式】for(let 変数名 of 配列名) {
　　処理；
　}

・for-of文の動作

①配列から要素を一つ取り出し、取り出せた場合は変数に代入して②に進む
　取り出せなかった場合はfor-of文の実行を終了する

②処理を実行して①に戻る

※要素の取り出しは、必ず配列の先頭から末尾に向かって順に行われる

77


```
01  'use strict';
02  const weights = [150, 250, 500];
03  console.log("for-of文");
04  for(let w of weights){                    //変数w：150→250→500
05    console.log("重さ" + w + "ｇまで");
06  }
07  console.log("for文");
08  for(let a = 0; a < weights.length; a++) {       //変数a：0→1→2
09    console.log("重さ" + weights[a] + "ｇまで");
10  }
11  console.log("while文");
12  let b = 0;
13  while(b < weights.length) {                     //変数b：0→1→2
14    console.log("重さ" + weights[b++] + "ｇまで");
15  }
```

・実行結果

```
for-of文
重さ150ｇまで
重さ250ｇまで
重さ500ｇまで
for文
重さ150ｇまで
重さ250ｇまで
重さ500ｇまで
while文
重さ150ｇまで
重さ250ｇまで
重さ500ｇまで
```

78

1.6　繰り返し構文

1.6.4　二重ループ

- 繰り返し構文のブロックに、別の繰り返し構文を入れる構造を
  ネスト（入れ子構造）という
- 二重ループの場合、外側の繰り返し構文の実行中に、内側の繰り返し構文が実行される

（例）

```
//外側のfor文
for(let d = 1; d <= 9 ; d++) {
    //内側のfor文
    for(let k = 1; k <= 9; k++) {
       console.log(d + ″ × ″ + k + ″ = ″ + (d * k));
    }
}
```



```
01  'use strict';
02  console.log("掛け算九九");
03  for(let d = 1; d <= 9; d++) {          //段の制御
04    for(let k = 1; k <= 9; k++) {        //数の制御
05      console.log(d + " × " + k + " = " + (d * k));
06    }
07  }
```

・実行結果

```
掛け算九九
1 × 1 = 1
1 × 2 = 2
1 × 3 = 3
1 × 4 = 4
1 × 5 = 5
      :
9 × 5 = 45
9 × 6 = 54
9 × 7 = 63
9 × 8 = 72
9 × 9 = 81
```

第１章　JavaScript基礎

１.６　繰り返し構文

１.６.５　break文

・繰り返し構文の実行中にbreak文を実行すると、
内側の繰り返し構文の実行を強制的に終了できる

（例）
```
//外側のfor文
for(let d = 1; d <= 4; d++) {
  //内側のfor文
  for(let k = 1; k <= 5; k++) {
    if(k == 4){
      break;  ←カウンタｋが４のときに内側のfor文を強制的に終了する
    }
    console.log(d + ″ × ″ + k + ″ = ″ + (d * k));
  }
}
```

81

第１章　JavaScript基礎

```
01  'use strict';
02  console.log("d × k の計算");
03  for(let d = 1; d <= 4; d++) {
04    for(let k = 1; k <= 5; k++) {
05      if(k == 4) {
06        break;        //内側のfor文を強制的に終了する
07      }
08      console.log(d + " × " + k + " = " + (d * k));
09    }
10  }
```

・実行結果

```
d × k の計算
1 × 1 = 1
1 × 2 = 2
1 × 3 = 3
2 × 1 = 2
2 × 2 = 4
2 × 3 = 6
3 × 1 = 3
3 × 2 = 6
3 × 3 = 9
4 × 1 = 4
4 × 2 = 8
4 × 3 = 12
```

82

１.６　繰り返し構文

１.６.６　ループのラベル

・繰り返し構文の実行中にラベル名を指定したbreak文を実行すると、
外側の繰り返し構文の実行を強制的に終了できる

（例）//外側のfor文

```
outer: for(let d = 1; d <= 4; d++) {
    //内側のfor文
    inner: for(let k = 1; k <= 5; k++) {
        if(k == 4){
            break outer;    ←カウンタｋが４のときに外側のfor文を強制的に終了する
        }
        console.log(d + ″ × ″ + k + ″ = ″ + (d * k));
    }
}
```

※ラベル「inner」は使用されないので省略可能


```
01  'use strict';
02  console.log("d × k の計算");
03  outer: for(let d = 1; d <= 4; d++) {
04      inner: for(let k = 1; k <= 5; k++) {
05          if(k == 4){
06              break outer;      //外側のfor文を強制的に終了する
07          }
08          console.log(d + " × " + k + " = " + (d * k));
09      }
10  }
```

・実行結果

```
d × k の計算
1 × 1 = 1
1 × 2 = 2
1 × 3 = 3
```

第１章　JavaScript基礎

１.７　ビルトイン関数

１.７.１　ビルトイン関数とは

- JavaScriptに組み込まれている関数群
- 特定のオブジェクトに依存しないため、グローバル関数とも呼ばれる
- 代表的なビルトイン関数

Number(string)	引数stringを数値に変換する
encodeURI(string)	引数stringを文字コードUTF-8の%xx形式にエンコードする
decodeURI(string)	引数stringを元の文字列にデコードする
eval(statements)	引数statementsをJavaScriptの構文として解釈・実行する

85

第１章　JavaScript基礎

１.７　ビルトイン関数

１.７.２　Number関数

- Number関数は、引数stringを数値型に変換する
- 数値に変換できない文字列を指定した場合は、
  NaN（Not-a-Number：数値ではない）が返される

```
（例）const str1 = ″10780″;                    //従量料金
      let billingAmount = Number(str1) + 1100;  //基本料金1100円
      console.log(″請求金額：″ + billingAmount + ″円″);
```

※「請求金額：11880円」と出力される

- 引数stringの先頭文字にあるプラス記号またはマイナス記号は符号として扱われる
- 引数stringの先頭または末尾にある複数の空白文字は、変換時に削除される

86

```
01  'use strict';
02  const str1 = "10780";
03  let billingAmount = Number(str1) + 1100;
04  console.log("請求金額：" + billingAmount + "円");
```

・**実行結果**

請求金額：11880円

1.7　ビルトイン関数

1.7.3　encodeURI関数

・encodeURI関数は、引数stringを文字コードUTF-8の%xx形式にエンコードする

（例）const str2 = `https://example.jp/テスト/index.html`;

let str3 = encodeURI(str2);　←エンコード

console.log(str3);

※「https://example.jp/%E3%83%86%E3%82%B9%E3%83%88/index.html」と出力される

・URI（Uniform Resource Identifier）はWeb上のファイルを識別するための情報で、URL（Uniform Resource Locator）とほぼ同じ

・全角文字の文字化けを防ぐ目的で利用される

・半角英数および記号（- _ . ! ~ * ' ( ) # ; , / ? : @ & = + $）は、エンコードの対象外となる

```
01  'use strict';
02  const str2 = "https://example.jp/テスト/index.html";
03  let str3 = encodeURI(str2);
04  console.log(str3);
```

・実行結果

https://example.jp/%E3%83%86%E3%82%B9%E3%83%88/index.html

１.７　ビルトイン関数

１.７.４　decodeURI関数

・decodeURI関数は、引数stringを元の文字列にデコードする

（例）const str4 = "https://example.jp/%E3%83%86%E3%82%B9%E3%83%88/index.html";

```
let str5 = decodeURI(str4);     ←デコード
console.log(str5);
```

※「https://example.jp/テスト/index.html」と出力される

```
01  'use strict';
02  const str4 = "https://example.jp/%E3%83%86%E3%82%B9%E3%83%88/index.html";
03  let str5 = decodeURI(str4);
04  console.log(str5);
```

・実行結果

https://example.jp/テスト/index.html

1.7　ビルトイン関数

1.7.5　eval関数

・eval関数は、引数statementsをJavaScriptの構文として解釈・実行する

（例）const str6 = ″>= 10″;　←本来は入力フォームなどから取得して代入

```
const code = ″const i = 20; if(i ″ + str6 + ″) { console.log('eval関数で実行'); }″;
eval(code);                    ←コードをJavaScriptの構文として解釈・実行
```

・ユーザが入力した文字列をコードに埋め込んで実行する目的で利用される

・ユーザが悪意のある文字列を入力するセキュリティリスクは無視できない

・eval関数でのコード実行は、通常の命令実行に比べて遅い

```
01  'use strict';
02  const str6 = ">= 10";          //本来は入力フォームなどから取得して代入
03  const code = "const i = 20; if(i " + str6 + ") { console.log('eval関数で実行'); }";
04  eval(code);
```

・実行結果
eval関数で実行

※変数codeの内容は、次のとおりである

```
const i = 20;
if(i >= 10) {
  console.log('eval関数で実行');
}
```


1.8　オブジェクトとメソッド

1.8.1　オブジェクトとは

・オブジェクトは、プロパティ（データ）とメソッド（データ処理）の集合体
（例）コンソールに出力する「console.log()」命令の場合
　　console … オブジェクト
　　log()　　… メソッド
・プロパティの利用は「オブジェクト名．プロパティ名」を記述
・メソッドの呼出しは「オブジェクト名．メソッド名()」を記述
・基本データ型のデータを扱うことができるオブジェクトが用意されている
　Numberオブジェクト　　←数値型
　Stringオブジェクト　　←文字列型
　Arrayオブジェクト　　←配列型

第１章　JavaScript基礎

1.8　オブジェクトとメソッド

1.8.2　Numberオブジェクト

- 数値（整数や実数）を一つだけ含むオブジェクト
- 記憶された数値または指定された数値を処理できる
- 代表的なメソッド

Number.parseInt(value)	文字列valueを整数に変換する
Number.parseFloat(value)	文字列valueを実数に変換する
変数.toLocaleString(locale)	変数の数値を文字列localeに合う文字列に変換する

※「変数.toLocaleString(locale)」は、変数の数値を含むNumberオブジェクトが自動的に生成され、そのオブジェクト内のtoLocaleString()メソッドが動作する

95

第１章　JavaScript基礎

1.8　オブジェクトとメソッド

1.8.3　toLocaleString()メソッド

- toLocaleStringメソッドは、変数の数値を文字列localeに合う文字列に変換する

【書式】変数.toLocaleString(locale)

※代表的な文字列locale

文字列locale	地域
ja-JP	日本
en-US	アメリカ
zh-CN	中国

（例）let billingAmount = 11880;
let billingAmountJPY = billingAmount.toLocaleString("ja-JP");
console.log("請求金額：" + billingAmountJPY + "円");

※「請求金額：11,880円」と出力される　←３桁区切りのカンマ記号が表示される

96

```
01  'use strict';
02  let billingAmount = 11880;
03  let billingAmountJPY = billingAmount.toLocaleString("ja-JP");
04  console.log("請求金額：" + billingAmountJPY + "円");
```

・実行結果

請求金額：11,880円

1.8　オブジェクトとメソッド

1.8.4　Stringオブジェクト

・文字列を一つだけ含むオブジェクト
・記憶された文字列または指定された文字列を処理できる
・代表的なメソッド

変数.split(delimiter)	変数の文字列に対して、引数delimiterの文字列で分割する
変数.replace(str1, str2)	変数の文字列に対して、引数str1の文字列を引数str2の文字列に置き換える
変数.replace(regex, str2)	変数の文字列に対して、引数regexの正規表現に一致する文字列を引数str2の文字列に置き換える

1.8　オブジェクトとメソッド

1.8.5　split()メソッド

・変数の文字列に対して、引数delimiterの文字列で分割する

【書式】変数.split(delimiter)

（例）　let preStr = ″1964/10/1″;

let postStr = preStr.split(″/″);　←分割後の各文字列が配列に記憶される

console.log(″分割後の文字列：″);

console.log(ps);

※「分割後の文字列：
1964、10、1」と出力される


```
01  'use strict';
02  let preStr = "1964/10/1";
03  let postStr = preStr.split("/");
04  console.log("分割前の文字列：" + preStr);
05  console.log("分割後の文字列：");
06  console.log(postStr);
```

・**実行結果**

分割前の文字列：1964/10/1

分割後の文字列：

(3)

1964

10

1

1.8　オブジェクトとメソッド

1.8.6　replace()メソッド

- 変数の文字列に対して、引数str1の文字列を引数str2の文字列に置き換える

  【書式】変数.replase(str1, str2)

  （例）　let preStr = ″ECMAScript″;

  let postStr = preStr.replace(″ECMA″, ″Java″);

  console.log(″置換後の文字列：″ + postStr);

  ※「置換後の文字列：JavaScript」と出力される
- 変数の文字列の先頭から順に、引数str1の文字列と一致する部分を探す
- 一致する部分が複数存在する場合は、最初に一致する部分だけ引数str2の文字列に置き換える


```
01  'use strict';
02  let preStr = "ECMAScript";
03  let postStr = preStr.replace("ECMA", "Java");
04  console.log("置換前の文字列：" + preStr);
05  console.log("置換後の文字列：" + postStr);
```

- **実行結果**

  置換前の文字列：ECMAScript

  置換後の文字列：JavaScript

1.8　オブジェクトとメソッド

1.8.7　正規表現を利用したreplace()メソッド

- 変数の文字列に対して、引数regexの正規表現に一致する文字列を引数str2の文字列に置き換える

【書式】変数.replase(regex, str2)

（例①）let preStr = ″ECMAScript, ECMAScript″;
　　let postStr = preStr.replace(/ECMA/g, ″Java″);
　　console.log(″置換後の文字列：″ + postStr);

※「置換後の文字列：JavaScript, JavaScript」と出力される

（例②）let billingAmountJPY = ″11,880″;
　　let billingAmount = billingAmountJPY.replace(/,/g, ″″);
　　console.log(″置換後の請求金額：″ + billingAmount);

※「置換後の請求金額：11880」と出力される　←カンマ記号は表示されない


```
01  'use strict';
02  let preStr = "ECMAScript, ECMAScript";
03  let postStr = preStr.replace(/ECMA/g, "Java");
04  console.log("置換前の文字列：" + preStr + "円");
05  console.log("置換後の文字列：" + postStr + "円");
06  let billingAmountJPY = "11,880";
07  let billingAmount = billingAmountJPY.replace(/,/g, "");
08  console.log("置換前の請求金額：" + billingAmountJPY + "円");
09  console.log("置換後の請求金額：" + billingAmount + "円");
```

- **実行結果**

置換前の文字列：ECMAScript, ECMAScript
置換後の文字列：JavaScript, JavaScript
置換前の請求金額：11,880円
置換後の請求金額：11880円

第１章　JavaScript基礎

1.8　オブジェクトとメソッド

1.8.7　正規表現を利用したreplace()メソッド

・正規表現の記述方法①

メタ文字	記号の意味	指定方法例	指定方法例の説明
[ ]	括弧内の任意の一文字 「-」で文字の範囲指定可能	[abc] [a-z]	a, b, cのいずれか１文字 a～zの１文字
[^]	括弧内の任意の１文字と不一致 「-」で文字の範囲を指定可能	[^ABC] [^1-9]	A, B, C以外 １～９以外
^	文字列の先頭	^A ^[1-9]	文字列の先頭がA 文字列の先頭が１～９のいずれか
$	文字列の末尾	A$ [A-Z]$	文字列の末尾がA 文字列の末尾がA～Zのいずれか
.	任意の１文字	. . .	任意の３文字の文字列（必ず3文字）
*	任意の文字列	*	任意の文字列（０文字以上）
+	直前の文字の１個以上の繰り返し	abc+	abcccのようにcが繰り返される文字列
?	直前の文字列の１個または０個	ABC?	ABCまたはAB

105

第１章　JavaScript基礎

1.8　オブジェクトとメソッド

1.8.7　正規表現を利用したreplace()メソッド

・正規表現の記述方法②

メタ文字	記号の意味	指定方法例	指定方法例の説明
{ }	直前の文字を括弧内の指定個で繰り返し	a{2} a{2, } a{2, 4}	aa aaまたはaaaまたはaaaaまたは･･･ aaまたはaaaまたはaaaa
( )	括弧内をグループ化	(abc){3}	abcabcabc
|	または	(abc)|(de)|f	abcまたはdeまたはf
¥w	半角英数、アンダースコア記号	¥w{4}	A12_3
¥d	０～９の数値	¥d{3}-¥d{4}	101-0052
¥s	タブ、改行、スペース	Java¥sScript	Java Script
¥	メタ文字の打ち消し	Java¥*Script	Java*Script

106

第１章　JavaScript基礎

1.8　オブジェクトとメソッド

1.8.8　Arrayオブジェクト

- 配列を一つだけ含むオブジェクト
- 記憶された配列の各要素を操作できる
- 代表的なメソッド

配列.unshift(value)	配列の先頭に引数valueを追加する
配列.shift()	配列の先頭の要素を削除する
配列.push(value)	配列の末尾に引数valueを追加する
配列.pop()	配列の末尾の要素を削除する
配列.splice(index, 0, value)	配列の引数index（添字）の位置に引数valueを追加する
配列.splice(index, cnt)	配列の引数index（添字）の位置から引数cntの個数分だけ要素を削除する

107

第１章　JavaScript基礎

1.8　オブジェクトとメソッド

1.8.9　unshift()メソッドとshift()メソッド

- unshift()メソッドは、配列の先頭に引数valueを追加する
- shift()メソッドは、配列の先頭の要素を削除する

【書式】配列.unshift(value)
　　配列.shift()

（例）
```
const pref1 = ["京都", "大阪", "東京"];
console.log(pref1);      ←「京都、大阪、東京」と出力される
pref1.unshift("兵庫");
console.log(pref1);      ←「兵庫、京都、大阪、東京」と出力される
pref1.shift();
console.log(pref1);      ←「京都、大阪、東京」と出力される
```

108

```
01  'use strict';
02  const pref1 = ["京都", "大阪", "東京"];
03  console.log(pref1);
04  pref1.unshift("兵庫");
05  console.log(pref1);
06  pref1.shift();
07  console.log(pref1);
```

・実行結果

```
(3)
  京都
  大阪
  東京
(4)
  兵庫
  京都
  大阪
  東京
(3)
  京都
  大阪
  東京
```


1.8　オブジェクトとメソッド

1.8.10　push()メソッドとpop()メソッド

・push()メソッドは、配列の末尾に引数valueを追加する
・pop()メソッドは、配列の末尾の要素を削除する

【書式】配列.push(value)
　　配列.pop()

（例）

```
const pref2 = [″京都″, ″大阪″, ″東京″];
console.log(pref2);      ←「京都、大阪、東京」と出力される
pref2.push(″兵庫″);
console.log(pref2);      ←「京都、大阪、東京、兵庫」と出力される
pref2.pop();
console.log(pref2);      ←「京都、大阪、東京」と出力される
```

```
01  'use strict';
02  const pref2 = ["京都", "大阪", "東京"];
03  console.log(pref2);
04  pref2.push("兵庫");
05  console.log(pref2);
06  pref2.pop();
07  console.log(pref2);
```

・実行結果

```
(3)
  京都
  大阪
  東京
(4)
  京都
  大阪
  東京
  兵庫
(3)
  京都
  大阪
  東京
```


1.8　オブジェクトとメソッド

1.8.11　splice()メソッド

・配列の引数index（添字）の位置に引数valueを追加する
・配列の引数index（添字）の位置から引数cntの個数分だけ要素を削除する

【書式】配列.splice(index, 0, value)　←第２引数「0」は“要素を削除しない”と指定
　　配列.splice(index, cnt)

（例）

```
const pref3 = [″京都″, ″大阪″, ″東京″];
console.log(pref3);      ←「京都、大阪、東京」と出力される
pref3.splice(2, 0, ″兵庫″);
console.log(pref3);      ←「京都、大阪、兵庫、東京」と出力される
pref3.splice(2, 1);
console.log(pref3);      ←「京都、大阪、東京」と出力される
```

```
01  'use strict';
02  const pref3 = ["京都", "大阪", "東京"];
03  console.log(pref3);
04  pref3.splice(2, 0, "兵庫");
05  console.log(pref3);
06  pref3.splice(2, 1);
07  console.log(pref3);
```

・実行結果

```
(3)
  京都
  大阪
  東京
(4)
  京都
  大阪
  兵庫
  東京
(3)
  京都
  大阪
  東京
```


1.9　ユーザ定義関数

1.9.1　ユーザ定義関数とは

- ユーザが独自に自分で定義した関数
- 複数の箇所に記述された同じ命令群をユーザ定義関数にまとめると、プログラムの保守性が高まる
- ユーザ定義関数の関数名、引数、戻り値の考え方、呼び出し方法は、ビルトイン関数と同じ
- 関数名の付け方のルール　　※変数名の付け方のルールと同じ

1文字目は、半角英字、アンダースコア記号、ドル記号が使用できる
２文字目以降は、数字も使用できる
予約語と同じ名前を変数名に付けることはできない

第１章　JavaScript基礎

1.9　ユーザ定義関数

1.9.2　ユーザ定義関数の作成

・ユーザ定義関数

【書式】function 関数名(引数) {　←引数は複数記述可能、省略可能

　　処理内容

　　return 戻り値；　←「return 戻り値；」の記述は省略可能

　}

（例）　//引数year（西暦）を和暦に変換して返す

　function japaneseCalendar(year) {　←「let year」と書いてはいけない

　　処理内容

　　return wareki；

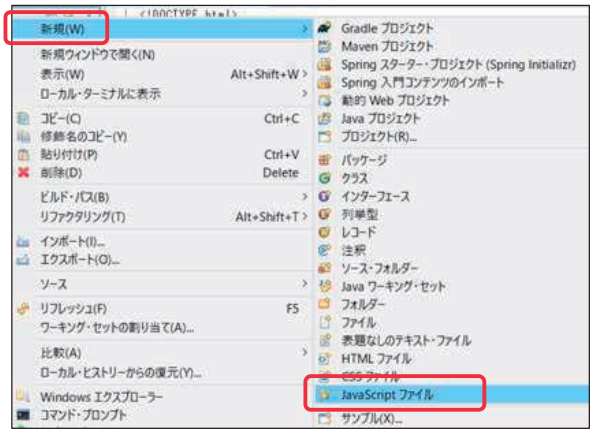
　}

第１章　JavaScript基礎

1.9　ユーザ定義関数

1.9.2　ユーザ定義関数の作成

・「script」フォルダを選択して、
メニューの「ファイル」→「新規」
→「JavaScriptファイル」を選択



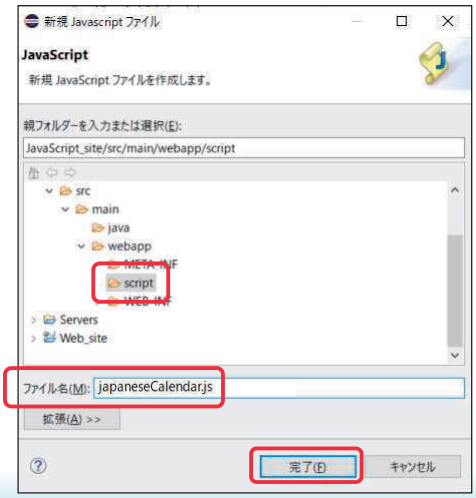
第１章　JavaScript基礎

1.9　ユーザ定義関数

1.9.2　ユーザ定義関数の作成

・保存フォルダ「script」
・ファイル名「japaneseCalendar.js」
・ファイル名を入力し「完了」ボタンをクリック

※ユーザ定義関数名とファイル名を一致させる
　決まりはない



117

第１章　JavaScript基礎

1.9　ユーザ定義関数

1.9.2　ユーザ定義関数の作成

・ユーザ定義関数「japaneseCalendar」の処理内容

①引数yearの西暦を区切り文字「/」で分割し、配列yearSplitに代入する

②もし要素yearSplit[0]の年が1912より小さいならば
　変数nengoに文字列「明治」を代入し、要素yearSplit[0]の年を1867減らす
そうでなく、もし要素yearSplit[0]の年が1926より小さいならば
　変数nengoに文字列「大正」を代入し、要素yearSplit[0]の年を1911減らす
そうでなく、もし要素yearSplit[0]の年が1989より小さいならば
　変数nengoに文字列「昭和」を代入し、要素yearSplit[0]の年を1925減らす
そうでなく、もし要素yearSplit[0]の年が2019より小さいならば
　変数nengoに文字列「平成」を代入し、要素yearSplit[0]の年を1988減らす
そうでないならば
　変数nengoに文字列「令和」を代入し、要素yearSplit[0]の年を2018減らす

③変数nengoの年号、要素yearSplit[0]の年、文字列「年」、要素yearSplit[1]の月、文字列「月」、要素yearSplit[2]の日、文字列「日」を文字列結合し、変数warekiに代入する

118

```
01  'use strict';
02  //引数year（西暦）を
03  //和暦に変換して返す
04  function japaneseCalendar(year) {
05    let nengo;
06    const yearSplit = year.split("/");
07    if(yearSplit[0] < 1912) {
08      nengo = "明治";
09      yearSplit[0] = yearSplit[0] - 1867;
10    } else if(yearSplit[0] < 1926) {
11      nengo = "大正";
12      yearSplit[0] = yearSplit[0] - 1911;
13    } else if(yearSplit[0] < 1989) {
14      nengo = "昭和";
15      yearSplit[0] = yearSplit[0] - 1925;
16    } else if(yearSplit[0] < 2019) {
17      nengo = "平成";
18      yearSplit[0] = yearSplit[0] - 1988;
19    } else {
20      nengo = "令和";
21      yearSplit[0] = yearSplit[0] - 2018;
22    }
23    let wareki = nengo + yearSplit[0] + "年"
24                 + yearSplit[1] + "月"
25                 + yearSplit[2] + "日";
26    return wareki;
27  }
```


1.9　ユーザ定義関数

1.9.3　ユーザ定義関数の呼び出し

・ユーザ定義関数の呼び出し

【書式】関数名(引数)

（例）　//japaneseCalendar関数を呼び出す

console.log(japaneseCalendar(″1964/10/1″) + ″　東海道新幹線開通″);

※「昭和39年10月1日　東海道新幹線開通」と出力される

```
05    <script src="script/japaneseCalendar.js"></script>
06    <script>
07      console.log(japaneseCalendar("1896/4/6") + "  第１回オリンピック開催");
08      console.log(japaneseCalendar("1919/6/28") + "  ベルサイユ講和条約調印");
09      console.log(japaneseCalendar("1964/10/1") + "  東海道新幹線開通");
10      console.log(japaneseCalendar("1995/11/23") + "  Windows95発売");
11      console.log(japaneseCalendar("2023/3/27") + "  国産初の量子コンピュータ稼働");
12    </script>
```

・実行結果

明治29年4月6日　第１回オリンピック開催
大正8年6月28日　ベルサイユ講和条約調印
昭和39年10月1日　東海道新幹線開通
平成7年11月23日　Windows95発売
令和5年3月27日　国産初の量子コンピュータ稼働

JavaScript
第２章
アルゴリズムの学習

【本章学習内容】

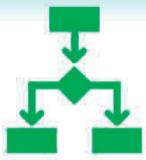
本章では、アルゴリズムについて学習します。

122

第２章　アルゴリズム

2.1　アルゴリズム

2.1.1　アルゴリズムとは



・手順１

野菜の皮をむく → 包丁で野菜をカットする → 肉を冷蔵庫から取り出す → 包丁で肉を一口大に切る → 炒め鍋に油をひき熱する → 素材を炒め鍋に投入する → 素材に味付けをする → しっかり火が通るまで炒める

・手順２

炒め鍋に油をひき熱する → 野菜の皮をむく → 包丁で野菜をカットする → 肉を冷蔵庫から取り出す → 包丁で肉を一口大に切る → 素材を炒め鍋に投入する → 素材に味付けをする → しっかり火が通るまで炒める

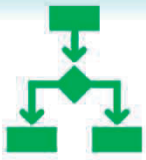
→この後、熱したフライパンはどうなる？

123

第2章　アルゴリズム

2.1　アルゴリズム

2.1.1　アルゴリズムとは



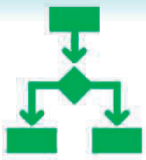
- アルゴリズムとは、問題を解決する方法や目標を達成するための手順のこと
- システム開発では、プログラムの処理手順を考える
- 同じ結果となるアルゴリズムでも、分かりやすさや処理効率に違いがある
- プログラミングの前に複数のアルゴリズムを検討し、より良いアルゴリズムを採用することが重要
- アルゴリズムを図示できるフローチャートなどを活用

124

第2章　アルゴリズム

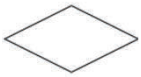
2.1　アルゴリズム

2.1.2　アルゴリズムの表現（フローチャート）



- 開始/終了記号
  フローチャートの開始点と終了点を示す

- 処理記号
  データ処理を示す



- 判断記号
  条件分岐を示す

- 定義済み処理
  既に定義されている別処理を呼び出すことを示す

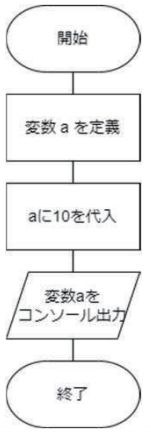
- 入出力記号
  外部装置（キーボード、ディスプレイなど）とのデータのやり取りを示す

125

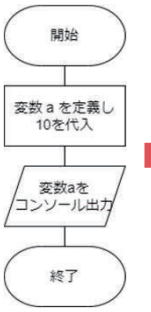
第2章　アルゴリズム

2.1　アルゴリズム

2.1.2　アルゴリズムの表現（フローチャート）




```
01　'use strict';
02　let a;
03　a = 10;
04　console.log(a);
```



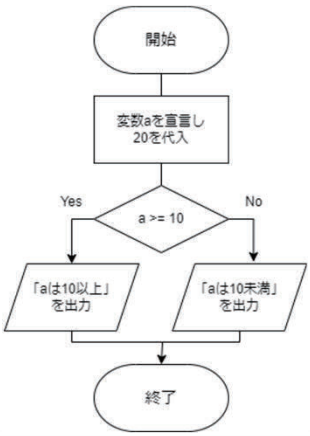

```
01　'use strict';
02　let a = 10;
03　console.log(a);
```

126

第2章　アルゴリズム

2.1　アルゴリズム

2.1.2　アルゴリズムの表現（フローチャート）



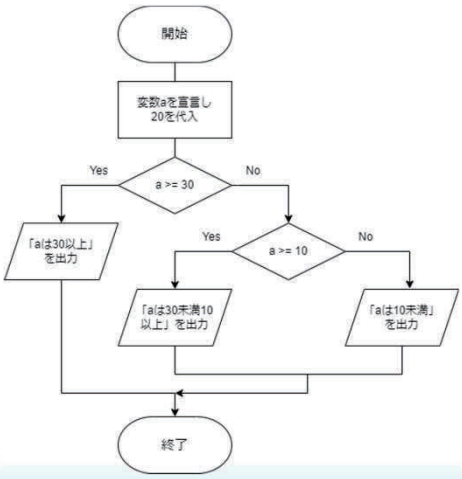

```
01　'use strict';
02　let a = 20;
03　if(a >= 10){
04　　console.log("aは10以上");
05　}else{
06　　console.log("aは10未満");
07　}
```

127

第2章　アルゴリズム

2.1　アルゴリズム

2.1.2　アルゴリズムの表現（フローチャート）



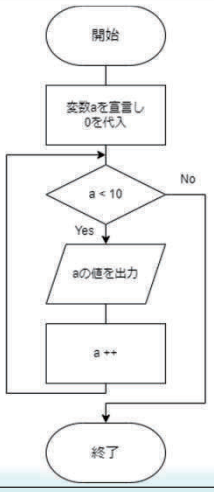

```
01  'use strict';
02  let a = 20;
03  if(a >= 30){
04      console.log("aは30以上");
05  }else if(a >= 10){
06      console.log("aは30未満10以上");
07  }else{
08      console.log("aは10未満")
09  }
```

128

第2章　アルゴリズム

2.1　アルゴリズム

2.1.2　アルゴリズムの表現（フローチャート）



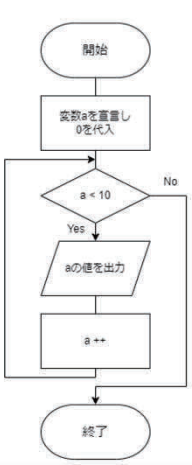

```
01  'use strict';
02  for(let a = 0; a < 10; a++){
03      console.log(a);
04  }
```

129

第2章　アルゴリズム

2.1　アルゴリズム

2.1.2　アルゴリズムの表現（フローチャート）



… ループ端記号（開始）
ここから繰り返し処理開始
記号内に繰り返しの継続条件
を記述

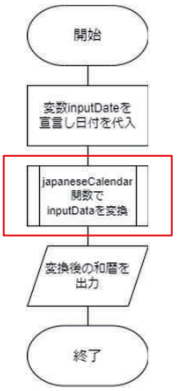
… ループ端記号（終了）
ここまでが繰り返し処理
記号内には何も記述しない

130

第2章　アルゴリズム

2.1　アルゴリズム

2.1.2　アルゴリズムの表現（フローチャート）



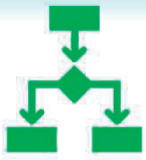

```
01　'use strict';
02　let inputDate = "2021/7/23";
03　let outputDate = japaneseCalendar(inputDate);
04　console.log(outputDate);
```

131

第２章　アルゴリズム

2.2　データ処理

2.2.1　データの交換



- 変数ａの値と変数ｂの値を交換するアルゴリズム
- アルゴリズムのポイント
  - 二つの変数の値はダイレクトに交換できない
  - アルゴリズムの最初の段階で処理「変数ａに変数ｂの値を代入」を記述した場合、変数ａの値は消えてしまう
  - 変数ｃを宣言して使用する

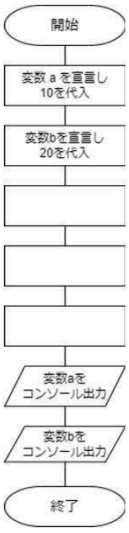
第２章　アルゴリズム

2.2　データ処理

2.2.1　データの交換

- 変数ａの値と変数ｂの値を、変数ｃを使用して交換するアルゴリズム
- 右のフローチャートの空欄を埋める

  演習時間：１分



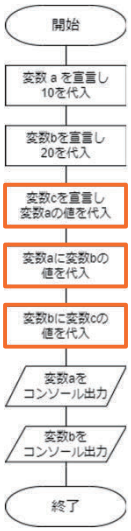
第２章　アルゴリズム

2.2　データ処理

2.2.1　データの交換

・解答例

オレンジ色の枠内を参照



第２章　アルゴリズム

2.2　データ処理

2.2.1　データの交換

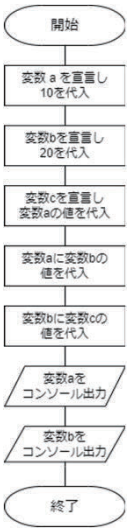
・右のフローチャートからJavaScriptのプログラムを作成する

「script」フォルダに
新規JavaScriptファイル「chap2.js」を作成

ファイル「JavaScript_SampleCheck.html」を開き、外部ファイル「script/chap2.js」を読み込むように修正

ファイル「chap2.js」にプログラムを入力

演習時間：５分



・解答例

```
01  'use strict';
02  console.log("データ交換");
03  let a = 10;
04  let b = 20;
05  let c = a;
06  a = b;
07  b = c;
08  console.log("変数ａ：" + a);
09  console.log("変数ｂ：" + b);
```

※ファイル「JavaScript_SampleCheck.html」の修正箇所

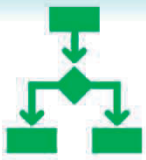
```
<script src="script/chap2.js"></script>
```

・実行結果

```
データ交換
変数ａ：20
変数ｂ：10
```


2.2　データ処理

2.2.2　データの合計



- 開始値から終了値までの合計を変数sumに求めるアルゴリズム
- アルゴリズムのポイント
  - フローチャートの書き方を工夫
  - アルゴリズムに
    処理「sum = sum + １」
    処理「sum = sum + ２」
    処理「sum = sum + ３」
    ：
    と記述した場合、保守性が低下する
  - 繰り返し構造を活用する

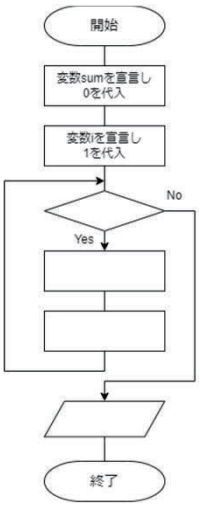
第２章　アルゴリズム

2.2　データ処理

2.2.2　データの合計

- 繰り返し構造により、１から10までの和を変数sumに求めるアルゴリズム
- 右のフローチャートの空欄を埋める

  演習時間：２分



138

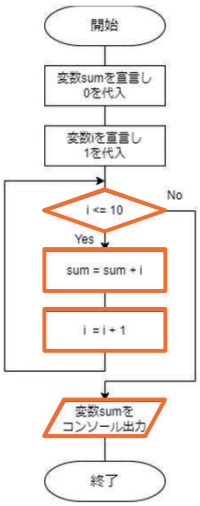
第２章　アルゴリズム

2.2　データ処理

2.2.2　データの合計

- 解答例

  オレンジ色の枠内を参照



139

第２章　アルゴリズム

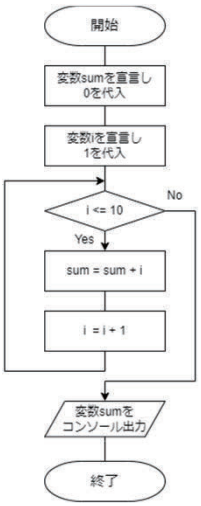
2.2　データ処理

2.2.2　データの合計

・右のフローチャートからJavaScriptのプログラムを作成する

ファイル「chap2.js」にプログラムを入力

演習時間：３分



140

第２章　アルゴリズム

・解答例

```
01  'use strict';
02  console.log("合計")
03  let sum = 0;
04  for(let i = 1; i <= 10; i++) {
05     sum = sum + i;
06  }
07  console.log("変数sum：" + sum);
```

・実行結果

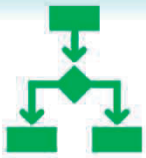
```
合計
変数sum：55
```

141

第２章　アルゴリズム

2.2　データ処理

2.2.3　データの個数



- 条件に合うデータの個数を求めるアルゴリズム
- アルゴリズムのポイント
  - 繰り返し構造で使用するカウンタ（変数）は、繰り返し制御用の変数である
  - カウンタ（変数）は、データの個数を求めるのに使用できない
  - 変数cntを宣言して使用する

142

第２章　アルゴリズム

2.2　データ処理

2.2.3　データの個数

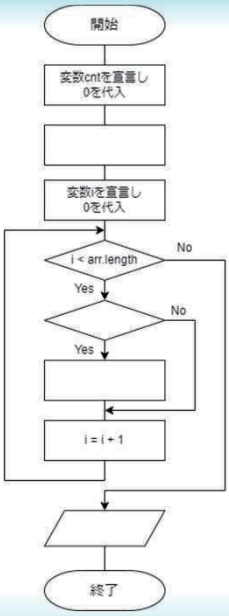
- 配列arrに格納されたデータのうち、偶数の個数を変数cntに求めるアルゴリズム

  配列arrの要素
  27, 39, 38, 21, 33, 42, 68, 47, 36

  「arr.length」は配列の要素数９を示す

  偶数判定は要素を２で割った余りを確認する
- 右のフローチャートの空欄を埋める

  演習時間：３分



143

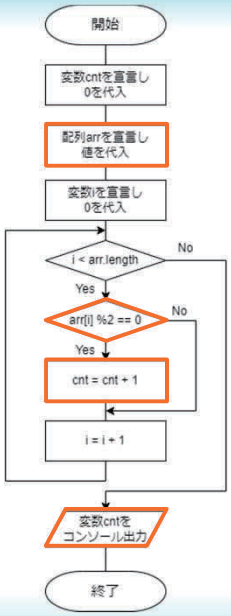
第２章　アルゴリズム

2.2　データ処理

2.2.3　データの個数

- 解答例

  オレンジ色の枠内を参照



144

第２章　アルゴリズム

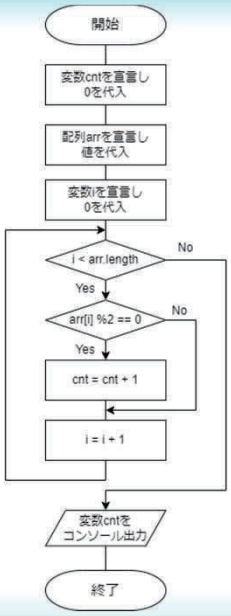
2.2　データ処理

2.2.3　データの個数

- 右のフローチャートからJavaScriptのプログラムを作成する

  ファイル「chap2.js」にプログラムを入力

  演習時間：４分



145

・解答例

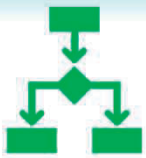
```
01  'use strict';
02  console.log("カウント")
03  const arr = [27, 39, 38, 21, 33, 42, 68, 47, 36];
04  let cnt = 0;
05  for(let i = 0; i < arr.length; i++) {
06    if(arr[i] % 2 == 0) {
07      cnt++;
08    }
09  }
10  console.log("変数cnt：" + cnt);
```

・実行結果

```
カウント
変数cnt：4
```


2.2　データ処理

2.2.4　データの平均



- すべてのデータを対象とした算術平均を変数avgに求めるアルゴリズム
- アルゴリズムのポイント
  - 平均は計算式「合計÷個数」で求める
  - 平均を求める前に、データの合計と個数を求めておく

第2章　アルゴリズム

2.2　データ処理

2.2.4　データの平均

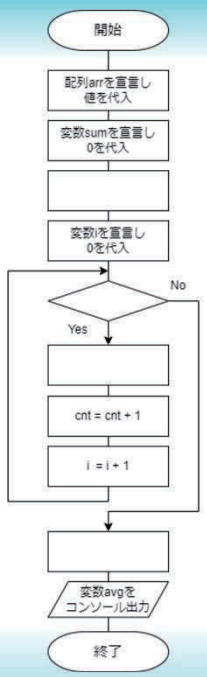
- 配列arrに格納されたデータの算術平均を変数avgに求めるアルゴリズム

  配列arrの要素
  27, 39, 38, 21, 33, 42, 68, 47, 36

  「arr.length」は配列の要素数９を示す

- 右のフローチャートの空欄を埋める

  演習時間：３分



148

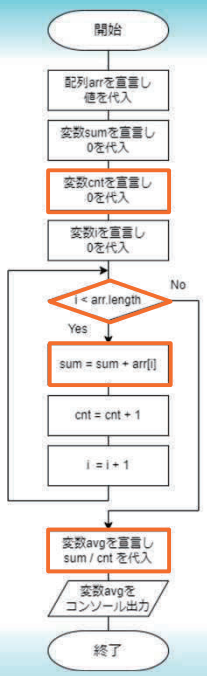
第2章　アルゴリズム

2.2　データ処理

2.2.4　データの平均

- 解答例

  オレンジ色の枠内を参照



149

第２章　アルゴリズム

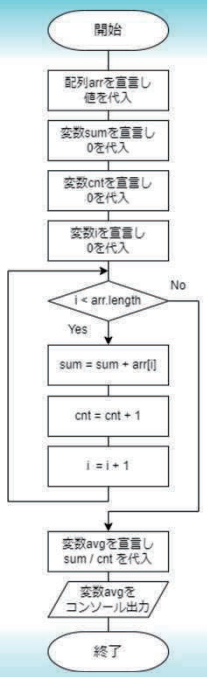
2.2　データ処理

2.2.4　データの平均

・右のフローチャートからJavaScriptのプログラムを作成する

ファイル「chap2.js」にプログラムを入力

演習時間：５分



150

第２章　アルゴリズム

・解答例

```
01  'use strict';
02  console.log("平均");
03  const arr = [27, 39, 38, 21, 33, 42, 68, 47, 36];
04  let sum = 0;
05  let cnt = 0;
06  for(let i = 0; i < arr.length; i++) {
07    sum = sum + arr[i];
08    cnt++;
09  }
10  let avg = sum / cnt;
11  console.log("変数avg：" + avg);
```

・実行結果

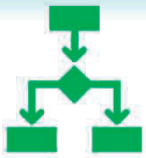
```
平均
変数avg：39
```

151

第２章　アルゴリズム

2.2　データ処理

2.2.5　データの最大・最小



- すべてのデータを比較して最大値または最小値を求めるアルゴリズム
- アルゴリズムのポイント
  - 繰り返し構造を使用して、最大値または最小値を効率よく求める
  - 変数valueを宣言して使用する

152

第２章　アルゴリズム

2.2　データ処理

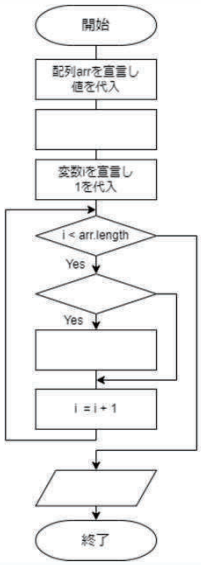
2.2.5　データの最大・最小

- 配列arrに格納されたデータの最大値を変数valueに求めるアルゴリズム

  配列arrの要素
  27, 39, 38, 21, 33, 42, 68, 47, 36

  「arr.length」は配列の要素数９を示す
- 右のフローチャートの空欄を埋める

  演習時間：３分



153

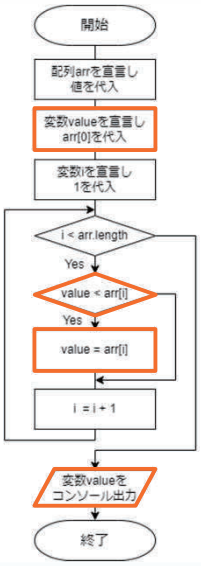
第２章　アルゴリズム

2.2　データ処理

2.2.5　データの最大・最小

・解答例

オレンジ色の枠内を参照



第２章　アルゴリズム

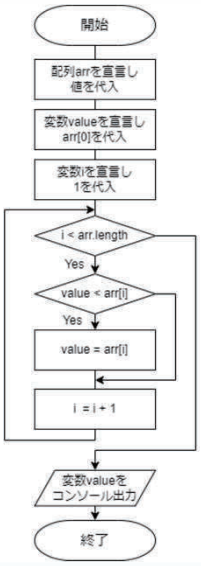
2.2　データ処理

2.2.5　データの最大・最小

・右のフローチャートからJavaScriptのプログラムを作成する

ファイル「chap2.js」にプログラムを入力

演習時間：５分



・解答例

```
01  'use strict';
02  console.log("最大値");
03  const arr = [27, 39, 38, 21, 33, 42, 68, 47, 36];
04  let value = arr[0];
05  for(let i = 1; i < arr.length; i++) {
06    if(value < arr[i]) {
07      value = arr[i];
08    }
09  }
10  console.log("変数value : " + value);
```

・実行結果

```
最大値
変数value : 68
```

※データの最小値を求める場合の修正箇所

```
02  console.log("最小値");
06    if(value > arr[i]) {
```

・実行結果

```
最小値
変数value : 21
```


2.3　探索（サーチ）

2.3.1　探索アルゴリズムとは

- データ群から探索値を探す処理手順
- 代表的なアルゴリズム

①線形探索法（リニアサーチ）

配列の先頭から順に、要素と探索値を比較する

②二分探索法（バイナリサーチ）

配列の探索範囲の中央に位置する要素と探索値を比較して、配列の探索範囲を狭める

③ハッシュ探索法

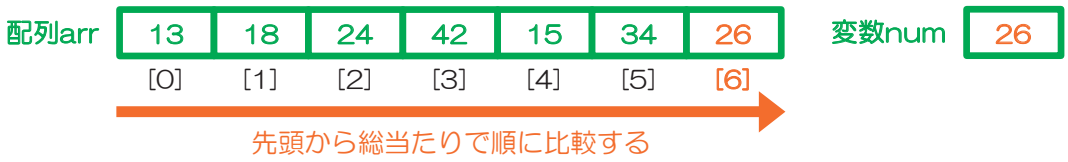
探索値から配列の添字（ハッシュ値）を求めて、その添字が示す要素と探索値を比較する

※本講座では、ハッシュ探索法の詳しい説明は行わない

第2章　アルゴリズム

2.3　探索（サーチ）

2.3.2　線形探索法（リニアサーチ）



- 配列に記憶されたデータの並び順は問われない
- 配列arrの先頭から順に、要素arr[i]と変数num（探索値）を比較する
- 変数num（探索値）と最初に一致する要素arr[i]の添字 i を求める
- 変数num（探索値）がどの要素とも一致しない場合は、「変数num（探索値）は見つからなかった」と結論づける

第2章　アルゴリズム

2.3　探索（サーチ）

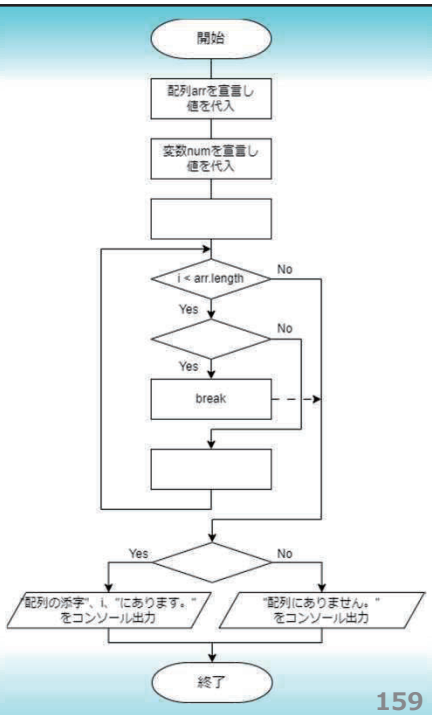
2.3.2　線形探索法（リニアサーチ）

- 配列arrに格納されたデータから、変数numを探すアルゴリズム

  配列arrの要素　　変数num
  13, 18, 24, 42, 15, 34, 26　　26

- 配列arrの先頭から順に、要素arr[i]と変数numを比較し、一致する要素arr[i]の添字iを求める
- 分岐条件「変数iが配列arrの要素数未満」を判断
  - Yesの場合は「配列の添字iにあります。」をコンソールに出力する
  - Noの場合は「配列にありません。」をコンソールに出力する
- 右のフローチャートの空欄を埋める

  演習時間：5分



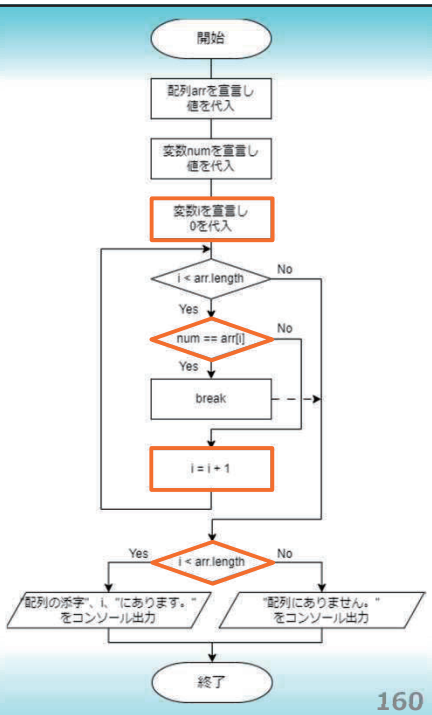
第2章　アルゴリズム

2.3　探索（サーチ）

2.3.2　線形探索法（リニアサーチ）

- 解答例

  オレンジ色の枠内を参照



160

第2章　アルゴリズム

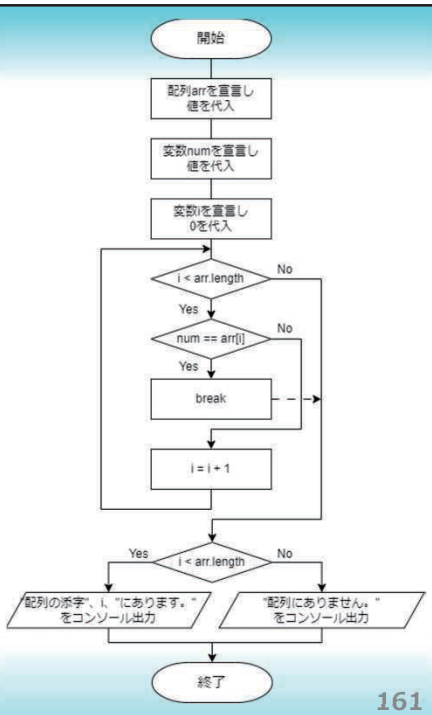
2.3　探索（サーチ）

2.3.2　線形探索法（リニアサーチ）

- 右のフローチャートからJavaScriptのプログラムを作成する

  ファイル「chap2.js」にプログラムを入力

  演習時間：5分



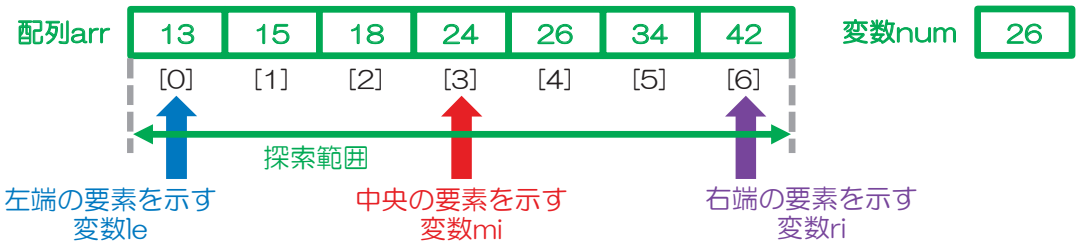
161

```
01  'use strict';
02  const arr = [13, 18, 24, 42, 15, 34, 26];
03  let num = 26;
04  let i;
05  for(i = 0; i < arr.length; i++) {
06    if(num == arr[i]){
07      break;
08    }
09  }
10  if(i < arr.length) {
11    console.log(num + "は配列の添字" + i + "にあります。");
12  }else{
13    console.log(num + "は配列にありません。");
14  }
```

・実行結果
26は配列の添字6にあります。

2.3　探索（サーチ）

2.3.3　二分探索法（バイナリサーチ）

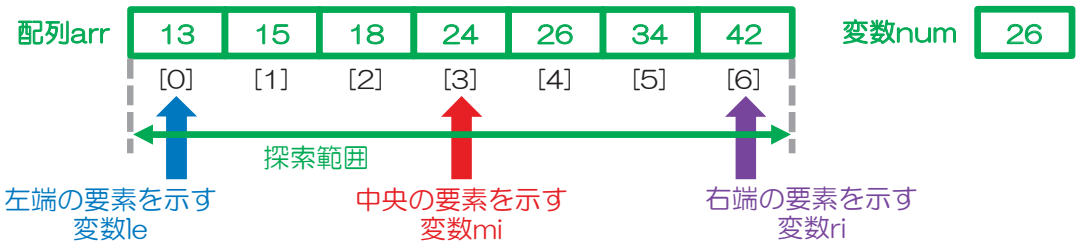


- 配列に記憶されたデータの並び順は、必ず昇順（小さい順）または降順（大きい順）でなければならない
- 左端の要素を示す変数leと右端の要素を示す変数riによって探索範囲が提示される
- 中央の要素を示す変数miは、計算式「(le + ri) / 2」の値とする

第2章　アルゴリズム

2.3　探索（サーチ）

2.3.3　二分探索法（バイナリサーチ）



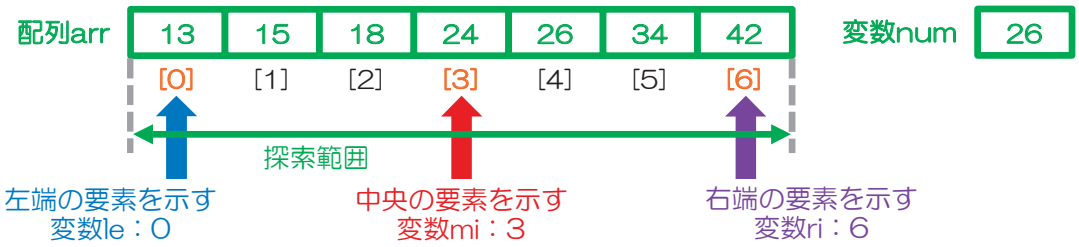
- 要素arr[mi]と変数num（探索値）を比較する
- 変数num（探索値）と最初に一致する要素arr[mi]の添字miを求める
- 左端の要素を示す変数leまたは右端の要素を示す変数riを更新し、中央の要素を示す変数miを再計算する
- 探索範囲を正しく提示できない場合は、「変数num（探索値）は見つからなかった」と結論づける

164

第2章　アルゴリズム

2.3　探索（サーチ）

2.3.3　二分探索法（バイナリサーチ）



①-1　変数leに0を代入、変数riに6を代入する

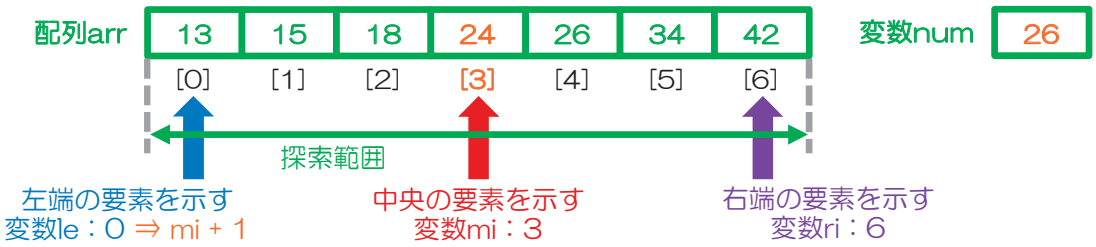
①-2　計算式「(le + ri) / 2」の値を求めて、変数miに代入する

165

第2章　アルゴリズム

2.3　探索（サーチ）

2.3.3　二分探索法（バイナリサーチ）



①-3　要素arr[mi]と変数numを比較する ⇒ 一致しない

①-4　左端の要素を示す変数leに計算式「mi + 1」の値を代入し、探索範囲を更新する

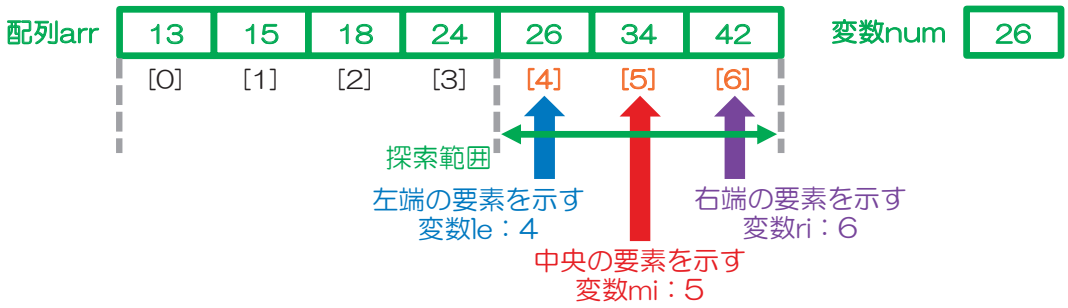
※データは昇順（小さい順）に並んでいるので、
変数numは中央の要素arr[mi]より右側に存在する可能性が高い

166

第2章　アルゴリズム

2.3　探索（サーチ）

2.3.3　二分探索法（バイナリサーチ）



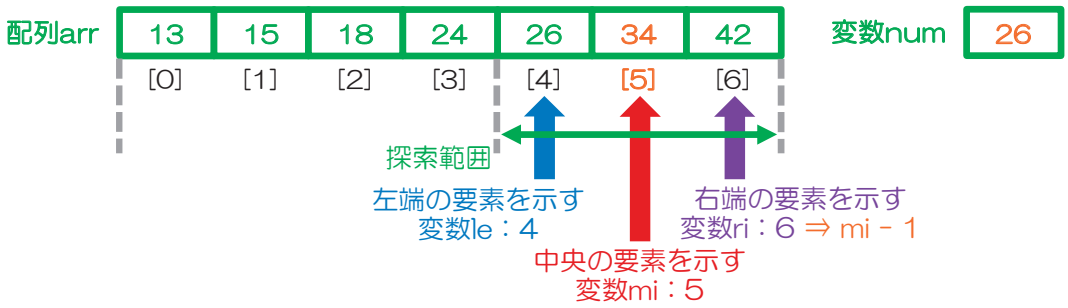
①-5　計算式「(le + ri) / 2」の値を求めて、変数miに代入する

167

第2章　アルゴリズム

2.3　探索（サーチ）

2.3.3　二分探索法（バイナリサーチ）



②-1　要素arr[mi]と変数numを比較する ⇒ 一致しない

②-2　右端の要素を示す変数riに計算式「mi － 1」の値を代入し、探索範囲を更新する

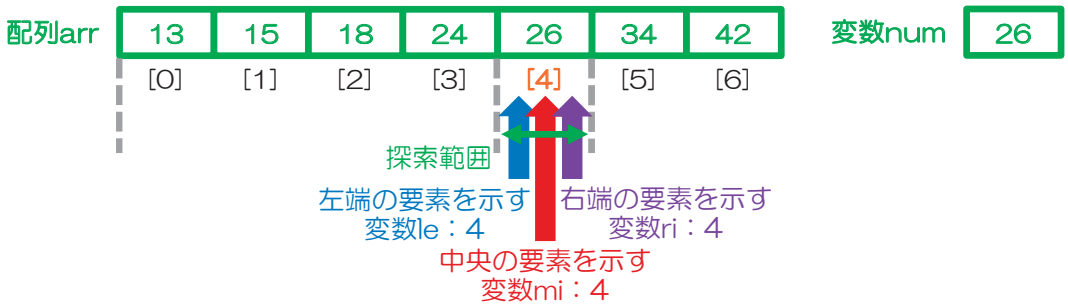
※データは昇順（小さい順）に並んでいるので、
変数numは中央の要素arr[mi]より左側に存在する可能性が高い

168

第2章　アルゴリズム

2.3　探索（サーチ）

2.3.3　二分探索法（バイナリサーチ）



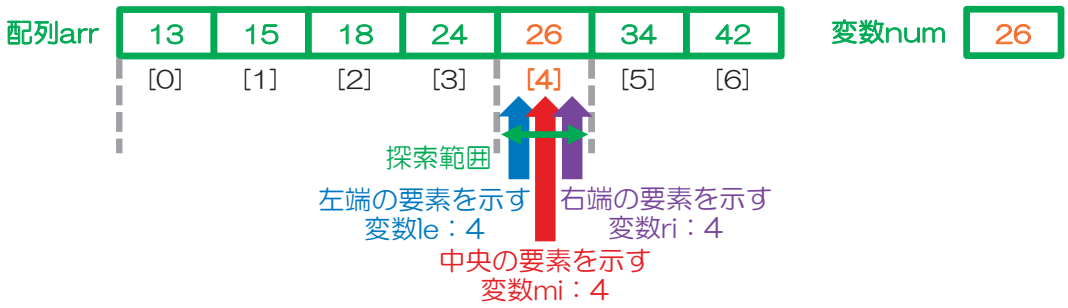
②-3　計算式「(le + ri) / 2」の値を求めて、変数miに代入する

169

第2章　アルゴリズム

2.3　探索（サーチ）

2.3.3　二分探索法（バイナリサーチ）



③-1　要素arr[mi]と変数numを比較する ⇒ 一致する「見つかった」

※一致しない場合は、探索範囲を正しく提示できないので
「見つからなかった」と結論づける

第2章　アルゴリズム

2.3　探索（サーチ）

2.3.3　二分探索法（バイナリサーチ）

- 配列arrに格納されたデータから、変数numを探すアルゴリズム

  配列arrの要素　　変数num
  13, 15, 18, 24, 26, 34, 42　　26

- 配列と変数を宣言する
  - 配列arr　：データ
  - 変数num：探索値
  - 変数le　：探索範囲の左端の要素を示す添字
  - 変数ri　：探索範囲の右端の要素を示す添字

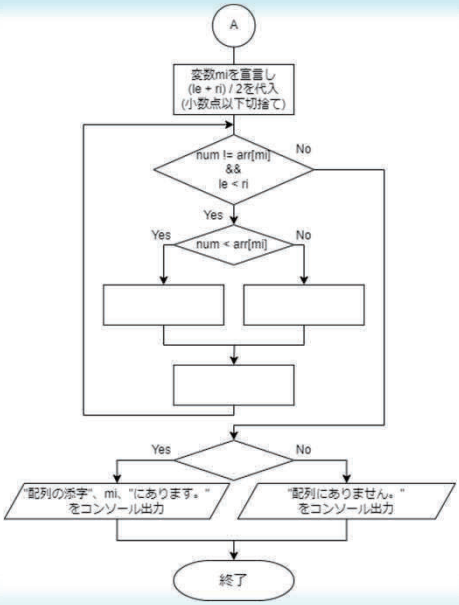
第2章　アルゴリズム

2.3　探索（サーチ）

2.3.3　二分探索法（バイナリサーチ）

- 変数miを宣言し、計算式「(le + ri) / 2」の小数部分を切り捨てて、整数部分を代入する
- 要素arr[mi]と変数numを比較し、一致する要素arr[mi]の添字miを求める
- 分岐条件「変数numが要素arr[mi]と等しい」を判断
  - Yesの場合は「配列の添字miにあります。」をコンソールに出力する
  - Noの場合は「配列にありません。」をコンソールに出力する
- 右のフローチャートの空欄を埋める

  演習時間：5分



172

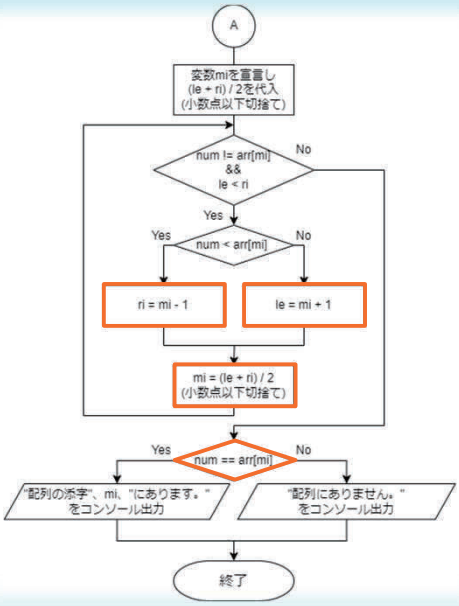
第2章　アルゴリズム

2.3　探索（サーチ）

2.3.3　二分探索法（バイナリサーチ）

- 解答例

  オレンジ色の枠内を参照



173

2.3　探索（サーチ）

2.3.3　二分探索法（バイナリサーチ）

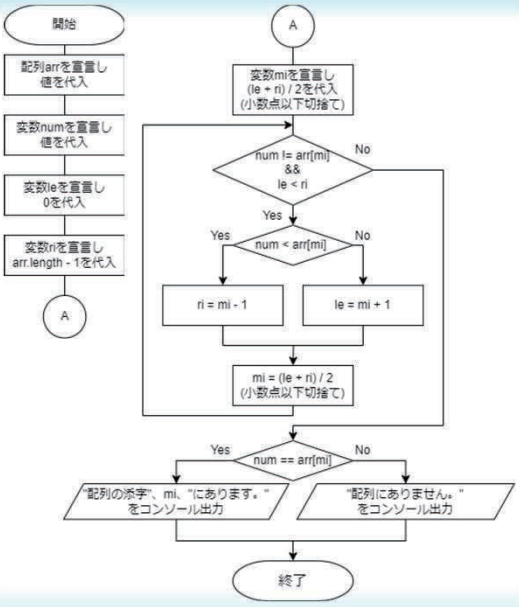
・右のフローチャートからJavaScriptのプログラムを作成する

ファイル「chap2.js」にプログラムを入力

演習時間：８分

※小数点以下の切捨ては、Math.floor()メソッドを使用する

（例）mi = Math.floor((le + ri) / 2);




```
01  'use strict';
02  const arr = [13, 15, 18, 24, 26, 34, 42];
03  let num = 26;
04  let le = 0;
05  let ri = arr.length - 1;
06  let mi = Math.floor((le + ri) / 2);
07  while(num != arr[mi] && le < ri) {
08    if(num < arr[mi]) {
09      ri = mi - 1;
10    } else {
11      le = mi + 1;
12    }
13    mi = Math.floor((le + ri) / 2);
14  }
15  if(num == arr[mi]) {
16    console.log(num
17              + "は配列の添字" + mi
18              + "にあります。");
19  } else {
20    console.log(num
21              + "は配列にありません。");
22  }
```

・実行結果

26は配列の添字4にあります。

第２章　アルゴリズム

2.4　並べ替え（ソート）

2.4.1　整列アルゴリズムとは

- データを昇順（小さい順）または降順（大きい順）に並べ替える処理手順
- 代表的なアルゴリズム
  - ①基本交換法（バブルソート）
    - 配列の未整列の要素群について、隣り合う要素を交換する
  - ②基本選択法（セレクションソート）
    - 配列の未整列の要素群から、１番目、２番目、…と順に選んで並べる
  - ③基本挿入法（インサーションソート）
    - 配列の未整列の要素群から要素を選んで、整列済みの要素群に挿入する
  - ④クイックソート
    - 配列の全要素を小さいグループと大きいグループに細かく分割する
  - ※本講座では、基本挿入法の詳しい説明は行わない

176

第２章　アルゴリズム

2.4　並べ替え（ソート）

2.4.2　基本交換法（バブルソート）

- 未整列の要素群（　　部分）について、隣り合う要素を交換する
  - 右図は、昇順（小さい順）に並べ替える場合を示す
  - 初期段階は、すべての要素を未整列の要素群とする
- 比較対象は、末尾の要素から先頭に向かって進める
- 隣りどうしの要素を比較し、並び順に合わせて交換する
- 先頭に最小値が移動する

177

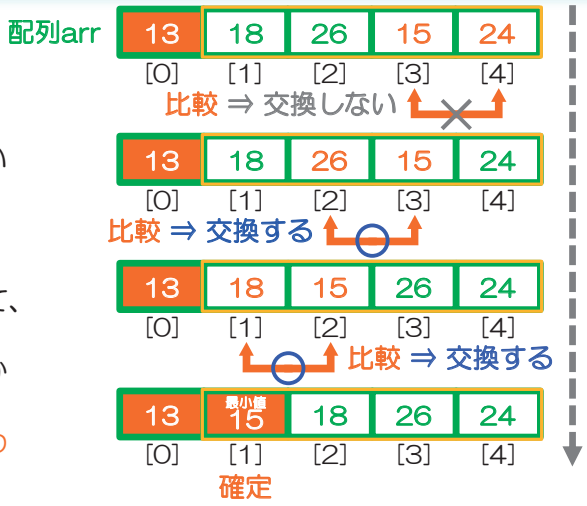
2.4　並べ替え（ソート）

2.4.2　基本交換法（バブルソート）

- 未整列の要素群から確定した要素を除いて、同じ操作を実行する

［再掲］

- 未整列の要素群（　　部分）について、隣り合う要素を交換する
- 比較対象は、末尾の要素から先頭に向かって進める
- 隣どうしの要素を比較し、並び順に合わせて交換する
- 先頭に最小値が移動する



2.4　並べ替え（ソート）

2.4.2　基本交換法（バブルソート）

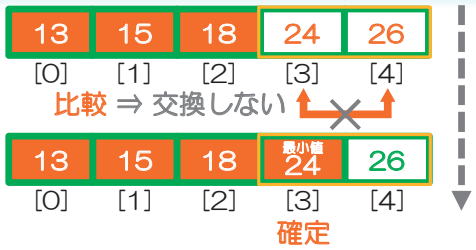
- 未整列の要素群から確定した要素を除いて、同じ操作を実行する

［再掲］

- 未整列の要素群（　　部分）について、隣り合う要素を交換する
- 比較対象は、末尾の要素から先頭に向かって進める
- 隣どうしの要素を比較し、並び順に合わせて交換する
- 先頭に最小値が移動する

2.4　並べ替え（ソート）

2.4.2　基本交換法（バブルソート）



- 未整列の要素群から確定した要素を除いて、同じ操作を実行する

[再掲]

- 未整列の要素群（　　部分）について、隣り合う要素を交換する
- 比較対象は、末尾の要素から先頭に向かって進める
- 隣どうしの要素を比較し、並び順に合わせて交換する
- 先頭に最小値が移動する

2.4　並べ替え（ソート）

2.4.2　基本交換法（バブルソート）

- 未整列の要素群から確定した要素を除く
  - 未整列の要素群の個数に注意する
- 未整列の要素群（　　部分）が残り１個となった時点で、すべての要素の位置が確定し、並べ替えが完了する
  - データが５個の場合、４個の要素の位置を確定させると並べ替えが完了する
  - データがｎ個の場合、n-1個の要素の位置を確定させると並べ替えが完了する

第２章　アルゴリズム

2.4　並べ替え（ソート）

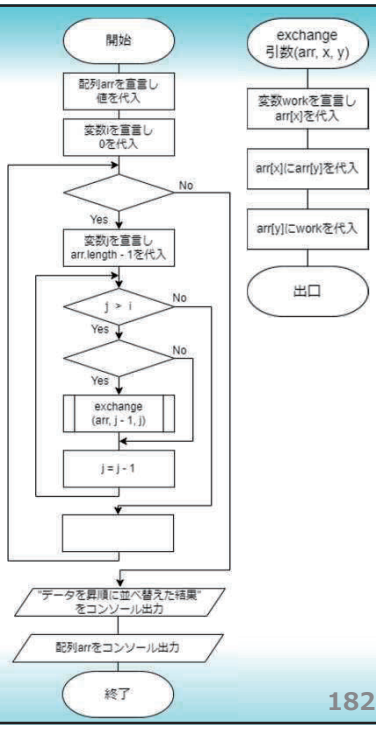
2.4.2　基本交換法（バブルソート）

- 配列arrに格納されたデータを昇順（小さい順）に並べ替えるアルゴリズム

  配列arrの要素
  18, 26, 15, 24, 13

- 外側の繰り返し処理は、データ５個のうち４個の位置を確定する
- 内側の繰り返し処理は、未整列の要素群の末尾の要素から先頭に向かって隣どうしの要素を比較し、未整列の要素群の先頭に最小値を移動する
- 定義済み処理「exchange」は、配列arrの添字xが示す要素と添字yが示す要素を交換する
- 右のフローチャートの空欄を埋める

  演習時間：５分



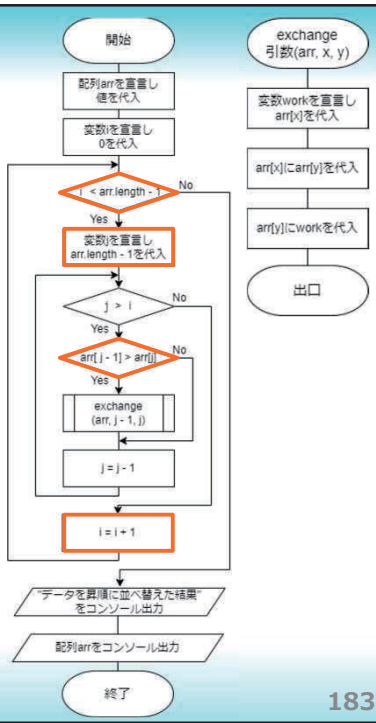
第２章　アルゴリズム

2.4　並べ替え（ソート）

2.4.2　基本交換法（バブルソート）

- 解答例

  オレンジ色の枠内を参照



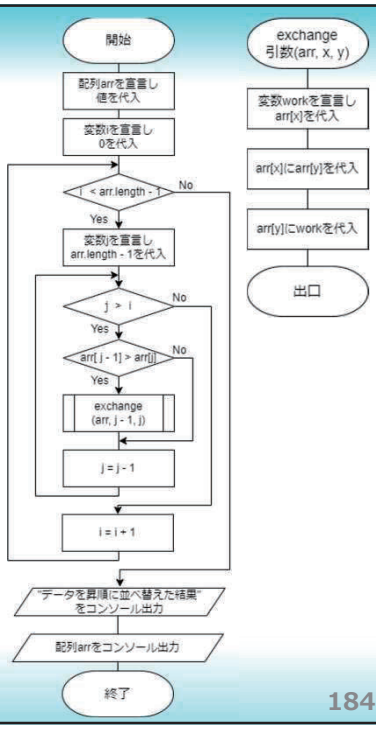
2.4　並べ替え（ソート）

2.4.2　基本交換法（バブルソート）

- 右のフローチャートからJavaScriptのプログラムを作成する

  ファイル「chap2.js」にプログラムを入力

  演習時間：７分




```
01  'use strict';
02  const arr = [18, 26, 15, 24, 13];
03  let i;
04  for(let i = 0; i < arr.length - 1; i++) {
05    for(let j = arr.length -1; j > i; j--) {
06      if(arr[j - 1] > arr[j]) {
07        exchange(arr, j -1 , j);
08      }
09    }
10  }
11  console.log("データを昇順に並べ替えた結果：");
12  console.log(arr);
13
14  function exchange(arr, x, y) {
15    let work = arr[x];
16    arr[x] = arr[y];
17    arr[y] = work;
18  }
```

・実行結果

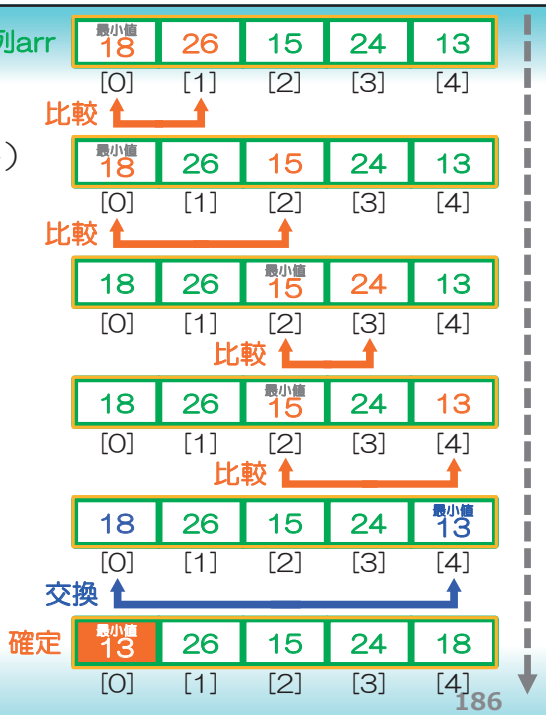
データを昇順に並べ替えた結果：
(5)
13
15
18
24
26

第2章　アルゴリズム

2.4　並べ替え（ソート）

2.4.3　基本選択法（セレクションソート）

- 未整列の要素群（　　部分）から、特定の値を選んで並べる
  - 右図は、昇順（小さい順）に並べ替える場合を示す
  - 初期段階は、すべての要素を未整列の要素群とする
- 比較対象は、未整列の要素群の先頭と、先頭の隣りの要素から末尾に向かって進める
- 未整列の要素群から最小値を探す
- 先頭に最小値を移動する（交換する）



186

第2章　アルゴリズム

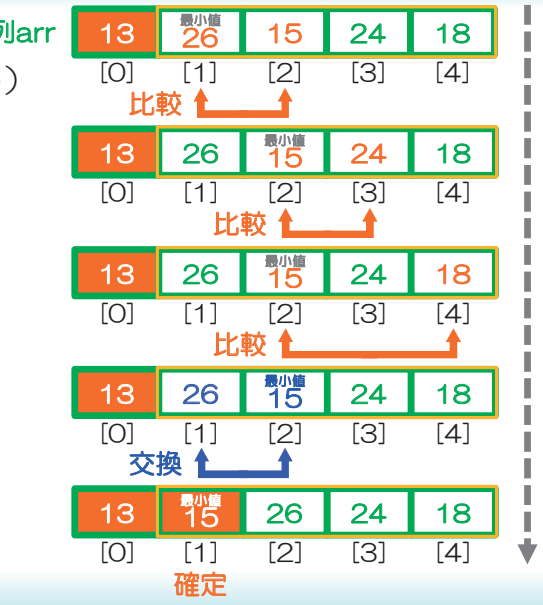
2.4　並べ替え（ソート）

2.4.3　基本選択法（セレクションソート）

- 未整列の要素群から確定した要素を除いて、同じ操作を実行する

［再掲］

- 未整列の要素群（　　部分）から、特定の値を選んで並べる
- 比較対象は、未整列の要素群の先頭と、先頭の隣から末尾に向かって進める
- 未整列の要素群から最小値を探す
- 先頭に最小値を移動する（交換する）



187

第2章　アルゴリズム

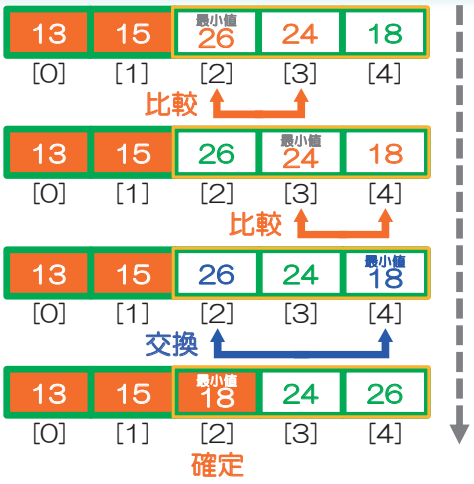
2.4　並べ替え（ソート）

2.4.3　基本選択法（セレクションソート）

- 未整列の要素群から確定した要素を除いて、同じ操作を実行する

［再掲］

- 未整列の要素群（　　部分）から、特定の値を選んで並べる
- 比較対象は、未整列の要素群の先頭と、先頭の隣から末尾に向かって進める
- 未整列の要素群から最小値を探す
- 先頭に最小値を移動する（交換する）



188

第2章　アルゴリズム

2.4　並べ替え（ソート）

2.4.3　基本選択法（セレクションソート）

- 未整列の要素群から確定した要素を除いて、同じ操作を実行する

［再掲］

- 未整列の要素群（　　部分）から、特定の値を選んで並べる
- 比較対象は、未整列の要素群の先頭と、先頭の隣から末尾に向かって進める
- 未整列の要素群から最小値を探す
- 先頭に最小値を移動する（交換する）

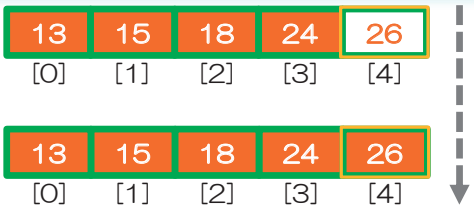
189

第２章　アルゴリズム

2.4　並べ替え（ソート）

2.4.3　基本選択法（セレクションソート）

配列arr



- 未整列の要素群から確定した要素を除く
  - 未整列の要素群の個数に注意する
- 未整列の要素群（　　部分）が残り１個となった時点で、すべての要素の位置が確定し、並べ替えが完了する
  - データが５個の場合、４個の要素の位置を確定させると並べ替えが完了する
  - データがｎ個の場合、n-1個の要素の位置を確定させると並べ替えが完了する

190

第２章　アルゴリズム

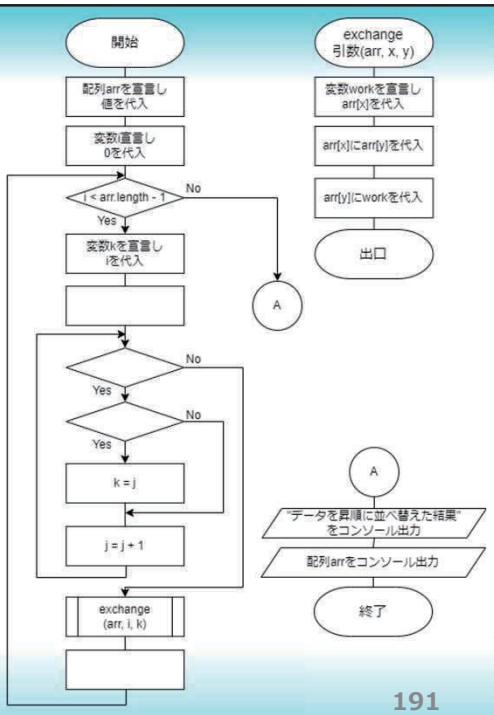
2.4　並べ替え（ソート）

2.4.3　基本選択法（セレクションソート）

- 配列arrに格納されたデータを昇順（小さい順）に並べ替えるアルゴリズム

  配列arrの要素
  18, 26, 15, 24, 13
- 外側の繰り返し処理は、データ５個のうち４個の位置を確定する
- 内側の繰り返し処理は、未整列の要素群の先頭の要素と、先頭の隣から末尾に向かって要素を順に比較する。未整列の要素群の先頭に最小値を移動する。
- 定義済み処理「exchange」は、配列arrの添字xが示す要素と添字yが示す要素を交換する
- 右のフローチャートの空欄を埋める

  演習時間：５分



191

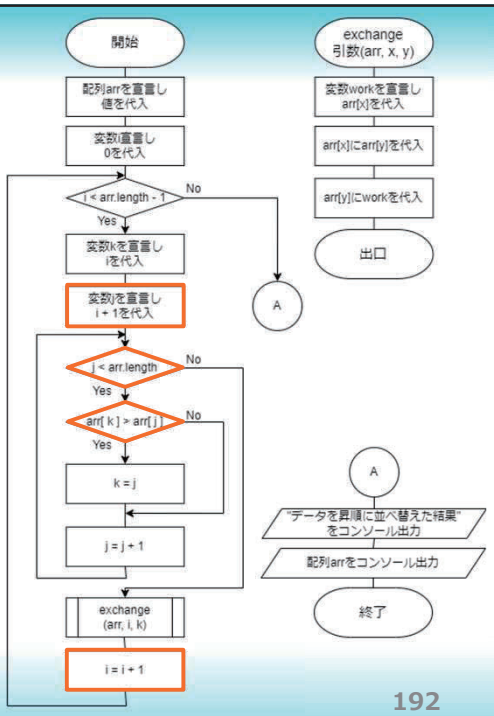
第２章　アルゴリズム

2.4　並べ替え（ソート）

2.4.3　基本選択法（セレクションソート）

・解答例

オレンジ色の枠内を参照



第２章　アルゴリズム

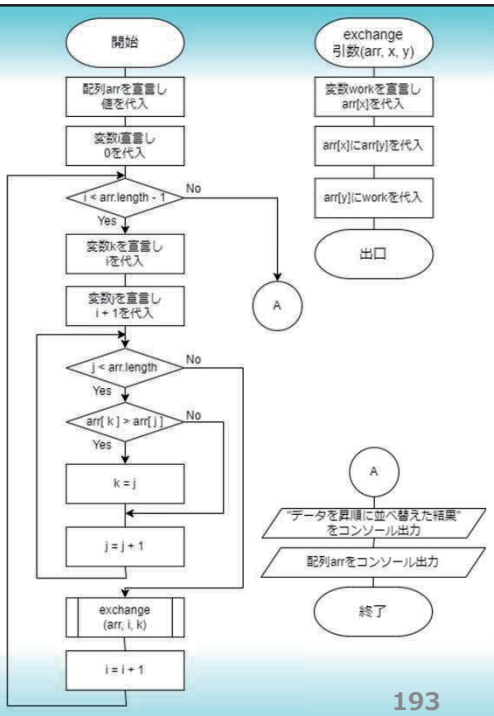
2.4　並べ替え（ソート）

2.4.3　基本選択法（セレクションソート）

・右のフローチャートからJavaScriptのプログラムを作成する

ファイル「chap2.js」にプログラムを入力

演習時間：７分



```
01  'use strict';
02  const arr = [18, 26, 15, 24, 13];
03  for(let i = 0; i < arr.length - 1; i++) {
04    let k = i;
05    for(let j = i + 1; j < arr.length; j++) {
06      if(arr[k] > arr[j]) {
07        k = j;
08      }
09    }
10    exchange(arr, i, k);
11  }
12  console.log("データを昇順に並べ替えた結果：");
13  console.log(arr);
14
15  function exchange(arr, x, y) {
16    let work = arr[x];
17    arr[x] = arr[y];
18    arr[y] = work;
19  }
```

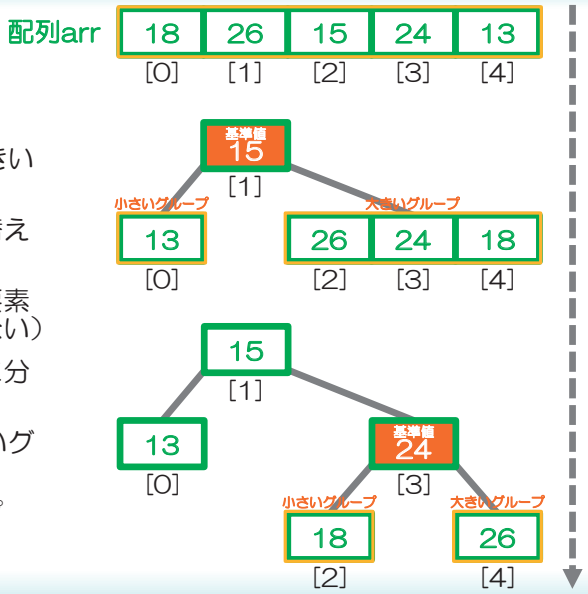
・実行結果

```
データを昇順に並べ替えた結果：
(5)
  13
  15
  18
  24
  26
```


2.4　並べ替え（ソート）

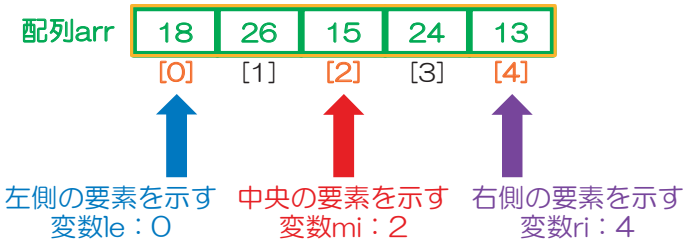
2.4.4　クイックソート

- 全要素を基準より小さいグループと大きいグループに分割する
  - 右図は、昇順（小さい順）に並べ替える場合を示す
  - 基準値を設定する（右図は中央の要素を基準値とするが、特に決まりはない）
  - 小さいグループと大きいグループに分割する際に、要素を交換する
- 各グループについて、基準値より小さいグループと大きいグループに分割する
  - 要素数が残り１個となったグループは分割を終了する



2.4　並べ替え（ソート）

2.4.4　クイックソート

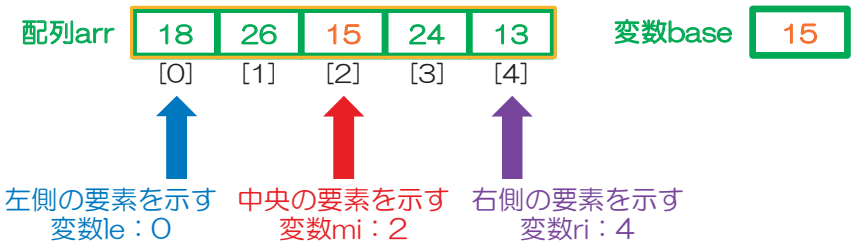


①-1　変数leに0を代入、変数riに4を代入する

①-2　計算式「(le + ri) / 2」の値を求めて、変数miに代入する（小数点以下切捨て）

2.4　並べ替え（ソート）

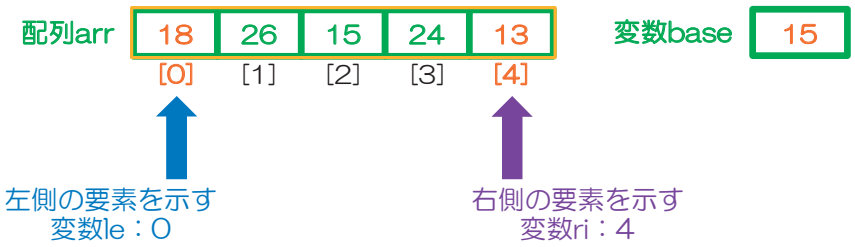
2.4.4　クイックソート



①-3　変数baseに要素arr[mi]を代入する

2.4　並べ替え（ソート）

2.4.4　クイックソート

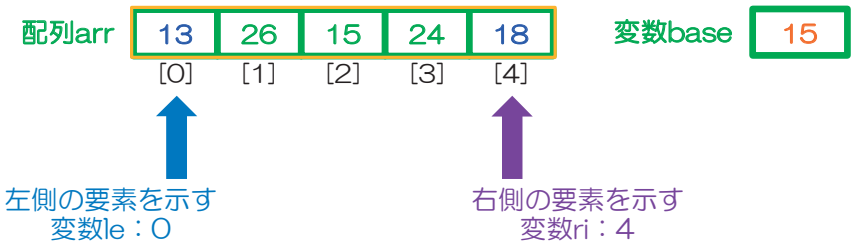


①-4　変数leを増やしながら、変数base以上の要素arr[le]を探す

①-5　変数riを減らしながら、変数base以下の要素arr[ri]を探す

2.4　並べ替え（ソート）

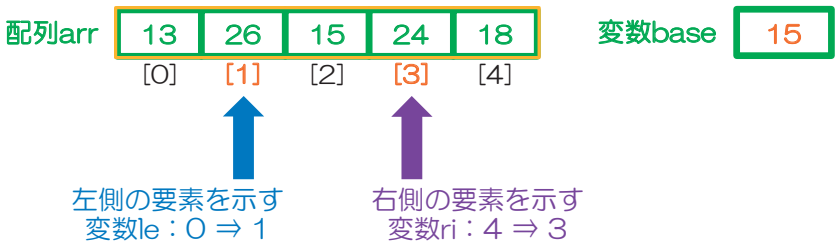
2.4.4　クイックソート



①-6　要素arr[le]と要素arr[ri]を交換する

2.4　並べ替え（ソート）

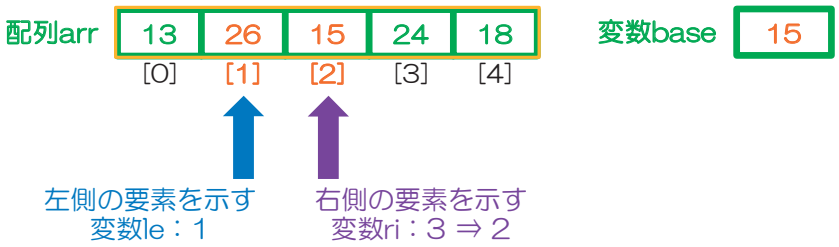
2.4.4　クイックソート



①-7　変数leを１増やす、変数riを１減らす

2.4　並べ替え（ソート）

2.4.4　クイックソート

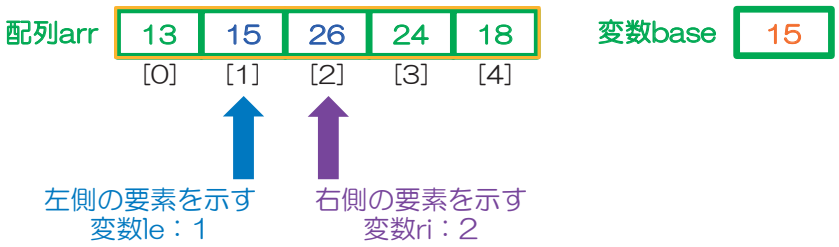


①-8　変数leを増やしながら、変数base以上の要素arr[le]を探す

①-9　変数riを減らしながら、変数base以下の要素arr[ri]を探す

2.4　並べ替え（ソート）

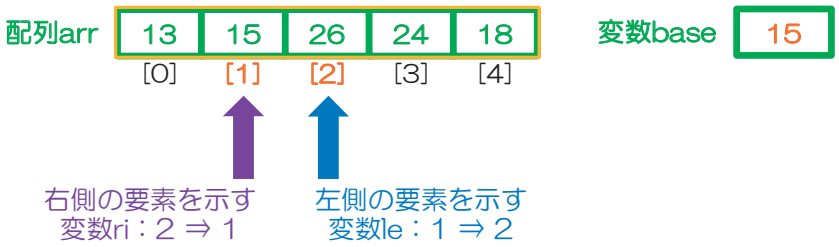
2.4.4　クイックソート



①-10　要素arr[le]と要素arr[ri]を交換する

2.4　並べ替え（ソート）

2.4.4　クイックソート

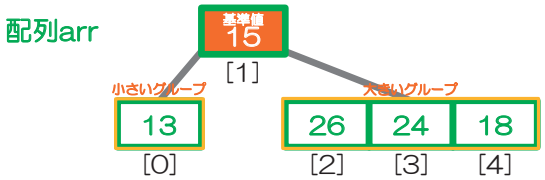


①-11　変数leを１増やす、変数riを１減らす

①-12　変数leと変数riを比較する ⇒ 左側の要素と右側の要素を正しく提示できない
⇒ 探して交換する作業を終了する

2.4　並べ替え（ソート）

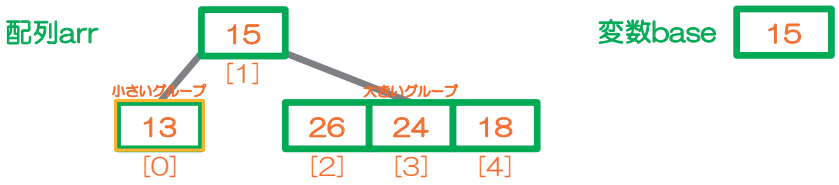
2.4.4　クイックソート



①-13　基準値より小さいグループと大きいグループができる

2.4　並べ替え（ソート）

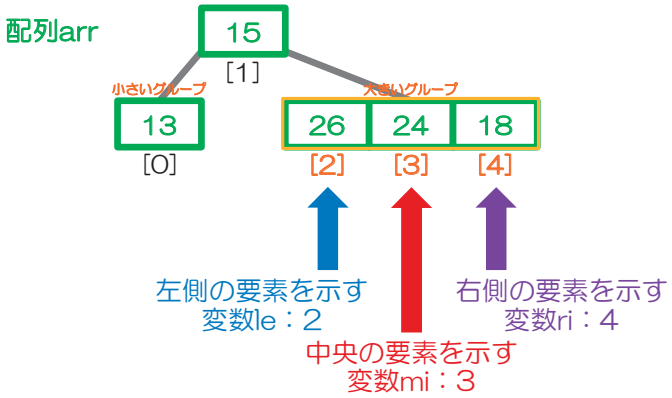
2.4.4　クイックソート



②小-1　要素数が残り１個となったグループは分割を終了する

2.4　並べ替え（ソート）

2.4.4　クイックソート



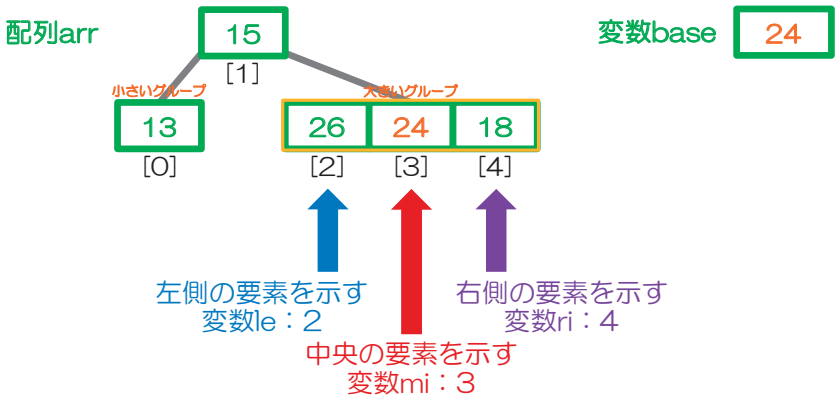
②大-1　変数leに2を代入、変数riに4を代入する

②大-2　計算式「(le + ri) / 2」の値を求めて、変数miに代入する

206

2.4　並べ替え（ソート）

2.4.4　クイックソート



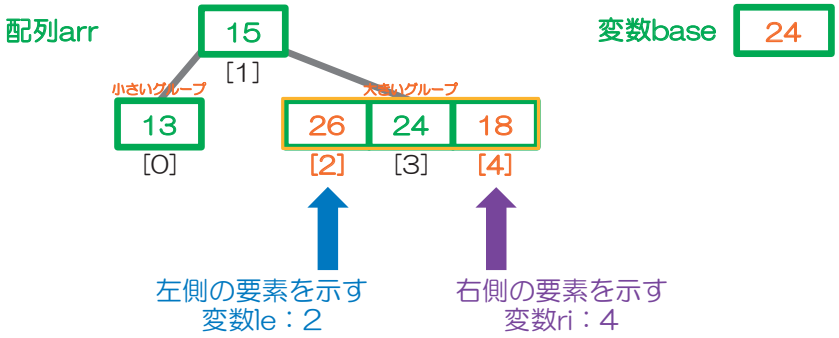
②大-3　変数baseに要素arr[mi]を代入する

207

第2章　アルゴリズム

2.4　並べ替え（ソート）

2.4.4　クイックソート



②大-4　変数leを増やしながら、変数base以上の要素arr[le]を探す

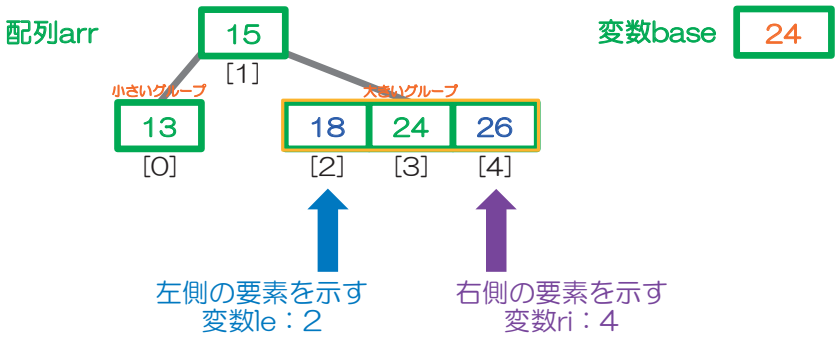
②大-5　変数riを減らしながら、変数base以下の要素arr[ri]を探す

208

第2章　アルゴリズム

2.4　並べ替え（ソート）

2.4.4　クイックソート

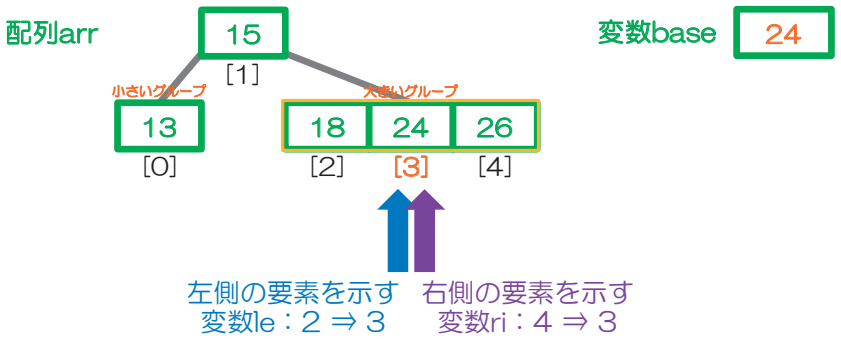


②大-6　要素arr[le]と要素arr[ri]を交換する

209

2.4　並べ替え（ソート）

2.4.4　クイックソート



②大-7　変数leを１増やす、変数riを１減らす

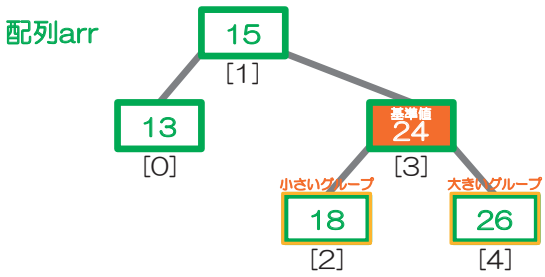
②大-8　変数leと変数riを比較する ⇒ 左側の要素と右側の要素を正しく提示できない

⇒ 探して交換する作業を終了する

210

2.4　並べ替え（ソート）

2.4.4　クイックソート

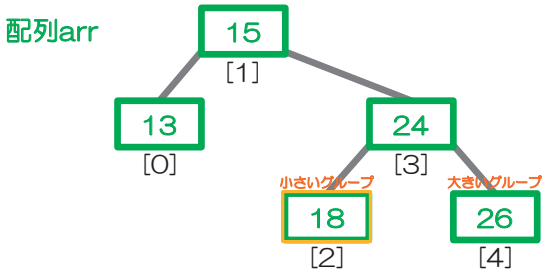


②大-9　基準値より小さいグループと大きいグループができる

211

2.4　並べ替え（ソート）

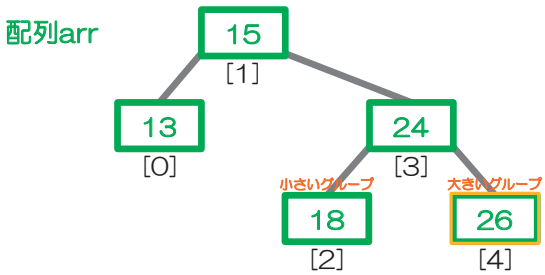
2.4.4　クイックソート



③小-1　要素数が残り１個となったグループは分割を終了する

2.4　並べ替え（ソート）

2.4.4　クイックソート



③大-1　要素数が残り１個となったグループは分割を終了する

2.4　並べ替え（ソート）

2.4.4　クイックソート

配列arr

13	15	18	24	26
[0]	[1]	[2]	[3]	[4]

並べ替え完了

④　すべてのグループの要素数が残り１個となった時点で**並べ替えが完了**する

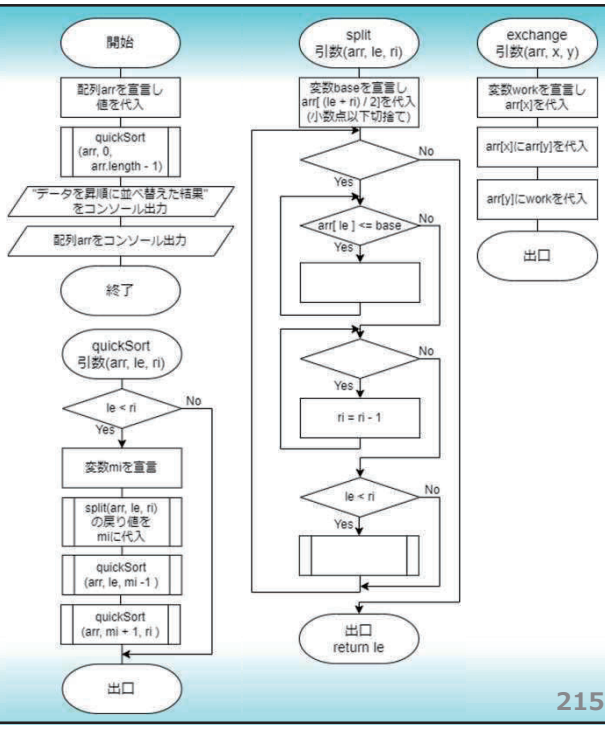
2.4　並べ替え（ソート）

2.4.4　クイックソート

- 配列arrに格納されたデータを昇順に並べ替えるアルゴリズム

  配列arrの要素
  18, 26, 15, 24 13
- 定義済み処理「quickSort」は、小さいグループと大きいグループで再帰呼出しを行う
- 定義済み処理「split」は、基準値より小さいグループと大きいグループに分割する
- 定義済み処理「exchange」は、要素どうしを交換する
- 右のフローチャートの空欄を埋める

  演習時間：5分



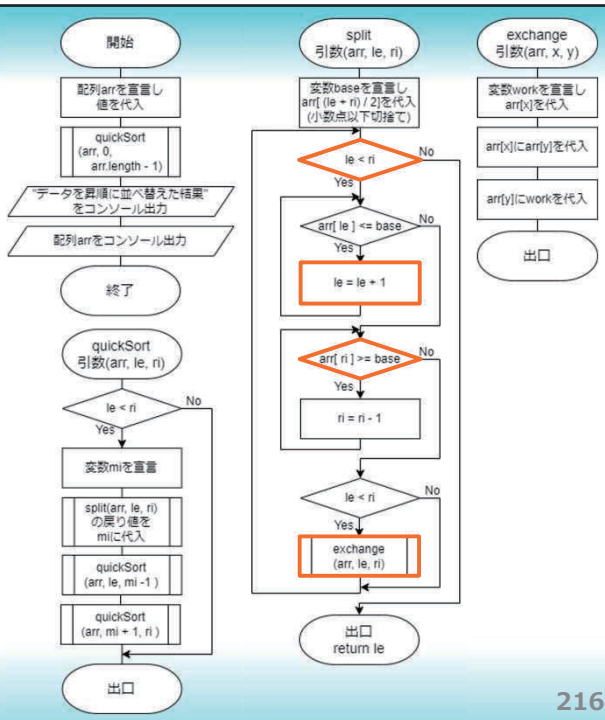
第２章　アルゴリズム

2.4　並べ替え（ソート）

2.4.4　クイックソート

- 解答例

  オレンジ色の枠内を参照



第２章　アルゴリズム

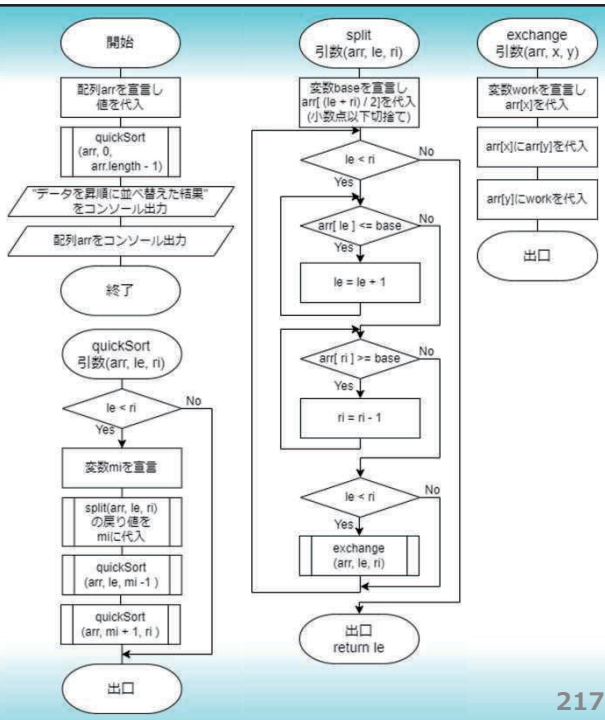
2.4　並べ替え（ソート）

2.4.4　クイックソート

- 右のフローチャートから
  JavaScriptのプログラムを作成
  する

  ファイル「chap2.js」にプロ
  グラムを入力

  演習時間：12分



```
01  'use strict';
02  const arr = [18, 26, 15, 24, 13];
03  quickSort(arr, 0, arr.length - 1);
04  console.log("データを昇順に並べ替えた結果：");
05  console.log(arr);
06  function quickSort(arr, le, ri) {
07    if(le < ri) {
08      let mi = split(arr, le, ri);
09      quickSort(arr, le, mi - 1);
10      quickSort(arr, mi + 1, ri);
11    }
12  }
13  function split(arr, le, ri) {
14    let base = arr[Math.floor((le + ri) / 2)];
15    while(le < ri) {
16      while(arr[le] < base) {
17        le++;
18      }
19      while(arr[ri] > base) {
20        ri--;
21      }
22      if(le < ri) {
23        exchange(arr, le, ri);
24      }
25    }
26    return le;
27  }
28  function exchange(arr, x, y) {
29    let work = arr[x];
30    arr[x] = arr[y];
31    arr[y] = work;
32  }
```

218

・実行結果

データを昇順に並べ替えた結果：

(5)

13

15

18

24

26

219

JavaScript
第３章
JavaScript応用

【本章学習内容】

本章では、HTMLタグ要素とフォーム操作について学習します。

220

第３章　JavaScript応用

３.１　HTMLタグ要素の取得

３.１.１　Webページを構成するオブジェクト

- JavaScriptでは「Webブラウザはオブジェクトである」という考え方を採用
- Webブラウザのオブジェクトには多数のプロパティやメソッドあり、特にプロパティは下位のオブジェクトと紐づいている
- 代表的なプロパティ

プロパティ名	オブジェクト名	対　象	概　要
window	Windowオブジェクト	Webブラウザ	ウィンドウに関する情報を管理
location	Locationオブジェクト	URL	URLに関する情報を管理
history	Historyオブジェクト	閲覧履歴	閲覧履歴に関する情報を管理
document	Documentオブジェクト	Webページ	Webページのタグ要素を管理

221

第３章　JavaScript応用

3.1　HTMLタグ要素の取得

3.1.1　Webページを構成するオブジェクト

- HTMLタグ要素をJavaScriptのプログラムから操作するには、Webページ内のHTMLタグ要素の取得が必要
- HTMLタグ要素を取得する方法

document.メソッド名()	概　要	取得形式
getElementById	引数と一致するid属性のHTMLタグ要素	単一
getElementsByClassName	引数と一致するclass属性のHTMLタグ要素	配列
getElementsByName	引数と一致するname属性のHTMLタグ要素	配列

222

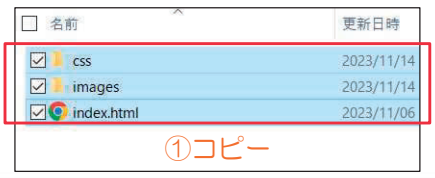
第３章　JavaScript応用

3.1　HTMLタグ要素の取得

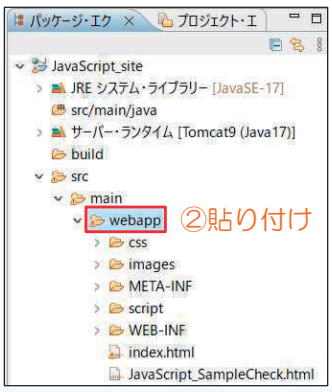
3.1.1　Webページを構成するオブジェクト

- 第３章の配布プログラム「JavaScript第３章.zip」を展開（解凍）
- 「index.html」ファイル、「css」フォルダ、「images」フォルダをコピー
- Eclipseの「webapp」フォルダ上で貼り付け

エクスプローラ



Eclipse



223

第３章　JavaScript応用

3.1　HTMLタグ要素の取得

3.1.1　Webページを構成するオブジェクト

- Eclipse上で「index.html」を選択
- メニューの「実行」→「実行」→「サーバで実行」を選択
- Webブラウザに「第３章 JavaScript応用」が表示される

※右のWebページ内のHTMLタグ要素に対して、JavaScriptのプログラムから操作する方法を学習する

Webページ

224

第３章　JavaScript応用

3.1　HTMLタグ要素の取得

3.1.1　Webページを構成するオブジェクト

- 「script」フォルダに新規JavaScriptファイル「chap3.js」を作成
- 「index.html」ファイルを開き、外部ファイル「script/chap3.js」を読み込むように修正
- 下記コードの９、10行目を追加

```
   :
06  <link href="css/reset.css" rel="stylesheet" type="text/css">
07  <link href="css/comon.css" rel="stylesheet" type="text/css">
08  <link href="css/main.css" rel="stylesheet" type="text/css">
09  <script src="script/chap3.js"></script>
10  <link rel="icon" href="data:,">
   :
```

225

3.1　HTMLタグ要素の取得

3.1.2　getElementByIdメソッド

・HTMLファイル内の、引数のid属性と一致するElementオブジェクトを返す
・Elementオブジェクトが見つからない場合は、予約語nullを返す

【書式】document.getElementById("id名")

（例）

HTMLファイル

```
<div class="lesson">
  <h3>HTML要素操作レッスン</h3>
  <p class="lessonDocs">この場所のHTMLタグ要素を操作します。</p>
  <p id="lesson1">ここはID「lesson1」が適用されている箇所です。</p>
  <p class="lesson2">ここはクラス「lesson2」が適用されている箇所です。</p>
</div>
```

JavaScriptプログラム

```
let elem = document.getElementById("lesson1");
```



```
01  'use strict';
02  //Webページの読み込みが完了した時点で動作するユーザ定義関数
03  window.onload = function() {
04    let elem = document.getElementById("lesson1");
05    console.log(elem);
06  }
```

・実行結果

```
<p id="lesson1">ここはID「lesson1」が適用されている箇所です。</p>
```

3.1　HTMLタグ要素の取得

3.1.3　getElementsByClassNameメソッド

・HTMLファイル内の、引数のclass属性と一致するElementオブジェクトの配列を返す
・Elementオブジェクトが見つからない場合は、要素数０の配列を返す

【書式】document.getElementsByClassName(″class名″)　　※メソッド名に注意

（例）

HTMLファイル

```
<div class="lesson">
  <h3>HTML要素操作レッスン</h3>
  <p class="lessonDocs">この場所のHTMLタグ要素を操作します。</p>
  <p id="lesson1">ここはID「lesson1」が適用されている箇所です。</p>
  <p class="lesson2">ここはクラス「lesson2」が適用されている箇所です。</p>
</div>
```

JavaScriptプログラム

```
let elems = document.getElementsByClassName("lesson2");
```



```
01  'use strict';
02  //Webページの読み込みが完了した時点で動作するユーザ定義関数
03  window.onload = function() {
04    let elems = document.getElementsByClassName("lesson2");
05    for(let elem of elems) {
06      console.log(elem);
07    }
08  }
```

・実行結果

```
<p class="lesson2">ここはクラス「lesson2」が適用されている箇所です。</p>
```

3.1　HTMLタグ要素の取得

3.1.4　getElementsByNameメソッド

・HTMLファイル内の、引数のname属性と一致するElementオブジェクトの配列を返す
・Elementオブジェクトが見つからない場合は、要素数０の配列を返す

【書式】document.getElementsByName(″name名″)　　※メソッド名に注意

（例）

HTMLファイル

```
<p>性別：
  <label><input type="radio" name="gender" value="男性">男性</label> 
  <label><input type="radio" name="gender" value="女性" checked>女性</label>
</p>
```

JavaScriptプログラム

```
let elems = document.getElementsByName("gender");
```



```
01  'use strict';
02  //Webページの読み込みが完了した時点で動作するユーザ定義関数
03  window.onload = function() {
04    let elems = document.getElementsByName("gender");
05    for(let elem of elems) {
06      console.log(elem);
07    }
08  }
```

・実行結果

```
<input type="radio" name="gender" value="男性">
<input type="radio" name="gender" value="女性" checked>
```

第３章　JavaScript応用

3.2　HTMLタグ要素のプロパティ

3.2.1　valueプロパティ

- 入力フォーム
  - テキストボックス（inputタグ）の入力値［送信値］
  - パスワードボックス（inputタグ）の入力値［送信値］
  - ラジオボタン（inputタグ）の選択値［送信値］
  - チェックボックス（inputタグ）の選択値［送信値］
  - ボタン（buttonタグ）の送信値
  - セレクトボックスの選択項目（optionタグ）の送信値　など
- その他
  - プログレスバー（progressタグ）やゲージ（meterタグ）の現在値
  - 順序付きリストの項目（liタグ）の表示番号　など

第３章　JavaScript応用

3.2　HTMLタグ要素の取得

3.2.1　valueプロパティ

- 入力フォームの入力値／選択値／送信値を格納

【書式】Elementオブジェクト.value

（例１）

HTMLファイル

```
<p>性別：
  <label><input type="radio" name="gender" value="男性">男性</label> 
  <label><input type="radio" name="gender" value="女性" checked>女性</label>
</p>
```

JavaScriptプログラム

```
let elems = document.getElementsByName("gender");
console.log(elems[0].value);
```

取得する

※「男性」と出力される

```
01  'use strict';
02  //Webページの読み込みが完了した時点で動作するユーザ定義関数
03  window.onload = function() {
04    let elems = document.getElementsByName("gender");
05    for(let elem of elems) {
06      console.log(elem.value);
07    }
08  }
```

・実行結果

男性

女性

3.2　HTMLタグ要素の取得

3.2.1　valueプロパティ

（例２）

HTMLファイル

```
<p>
  <label>名前：
    <input type="text" name="name" value=""
           placeholder="名前を入力してください">
  </label>
</p>
```

JavaScriptプログラム

```
let elems = document.getElementsByName("name");
elems[0].value = "聖徳太子";
```

更新する

※Webページの「氏名」テキスト入力ボックスに「聖徳太子」と表示される

```
01  'use strict';
02  //Webページの読み込みが完了した時点で動作するユーザ定義関数
03  window.onload = function() {
04    let elems = document.getElementsByName("name");
05    elems[0].value = "聖徳太子";
06  }
```

・実行結果

［実行前］ 名前：名前を入力してください

［実行後］ 名前：聖徳太子

3.2　HTMLタグ要素のプロパティ

3.2.2　innerHTMLプロパティ

・開始タグと終了タグで囲まれた部分を格納

【書式】Elementオブジェクト.innerHTML

（例１）

HTMLファイル

```
<p class="lessonDocs">この場所のHTMLタグ要素を操作します。</p>
<p id="lesson1">ここはID「lesson1」が適用されている箇所です。</p>
<p class="lesson2">ここはクラス「lesson2」が適用されている箇所です。</p>
```

JavaScriptプログラム

```
let elem = document.getElementById("lesson1");
console.log(elem.innerHTML);
```

取得する

※「ここはID「lesson1」が適用されている箇所です。」と出力される

```
01  'use strict';
02  //Webページの読み込みが完了した時点で動作するユーザ定義関数
03  window.onload = function() {
04    let elem = document.getElementById("lesson1");
05    console.log(elem.innerHTML);
06  }
```

・実行結果
ここはID「lesson1」が適用されている箇所です。

3.2　HTMLタグ要素のプロパティ

3.2.2　innerHTMLプロパティ

（例２）

HTMLファイル

```
<p class="lessonDocs">この場所のHTMLタグ要素を操作します。</p>
<p id="lesson1">ここはID「lesson1」が適用されている箇所です。</p>
<p class="lesson2">ここはクラス「lesson2」が適用されている箇所です。</p>
```

JavaScriptプログラム

```
let elems = document.getElementsByClassName("lesson2");
elems[0].innerHTML = "クラス「lesson2」の要素を書き換えました。";
```

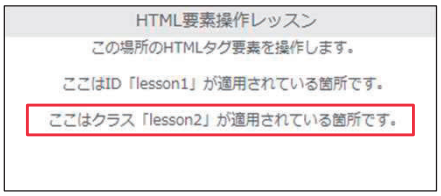
更新する

※Webページの段落に「クラス「lesson2」の要素を書き換えました。」と表示される

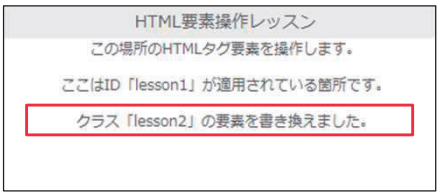
```
01  'use strict';
02  //Webページの読み込みが完了した時点で動作するユーザ定義関数
03  window.onload = function() {
04    let elems = document.getElementsByClassName("lesson2");
05    elems[0].innerHTML = "クラス「lesson2」の要素を書き換えました。";
06  }
```

・実行結果

［実行前］



［実行後］



3.2　HTMLタグ要素のプロパティ

3.2.3　srcプロパティ

・埋め込む画像（imgタグ）のURLを格納

【書式】Elementオブジェクト.src

（例）

HTMLファイル

```
<div class="header_left">
  <h1>
    <img src="images/chap3logo.png" alt="ロゴ" id="logo">
  </h1>
</div>
```

JavaScriptプログラム

```
let elem = document.getElementById("logo");
elem.src = "images/chap3logo2.png";
```

更新する

※Webページの画像が更新される

```
01  'use strict';
02  //Webページの読み込みが完了した時点で動作するユーザ定義関数
03  window.onload = function() {
04    let elem = document.getElementById("logo");
05    elem.src = "images/chap3logo2.png";
06  }
```

・実行結果

［実行前］

［実行後］

242

第３章　JavaScript応用

3.2　HTMLタグ要素のプロパティ

3.2.4　rowsプロパティ

・rowsプロパティは、テーブル（tableタグ）の行（trタグ）

【書式】Elementオブジェクト.rows　→テーブルの全行（配列形式）
Elementオブジェクト.rows.length　→テーブルの行数

（例）

HTMLファイルの表示結果

注文金額	本州・四国	北海道・九州	沖縄
3,000円以上	500円	700円	950円
5,000円以上	700円	850円	
10,000円以上	1,500円		
備考	購入金額が¥20,000円を超える場合は、 送料が変わる場合がありますのでお問合せください。		

JavaScriptプログラム

```
let elem = document.getElementById("postage");
console.log("行数：" + elem.rows.length);
```

行数を取得する

※「行数：5」が出力される

243

3.2　HTMLタグ要素のプロパティ

3.2.4　rowsプロパティ

・cellsプロパティは、テーブル（tableタグ）の特定行（trタグ）の列（thタグ、tdタグ）

【書式】Elementオブジェクト.rows[行番号].cells　→行の全列（配列形式）
Elementオブジェクト.rows[行番号].cells.length　→行の列数

（例）

HTMLファイルの表示結果

注文金額	本州・四国	北海道・九州	沖縄
3,000円以上	500円	700円	950円
5,000円以上	700円	850円	
10,000円以上	1,500円		
備考	購入金額が¥20,000円を超える場合は、 送料が変わる場合がありますのでお問合せください。		

JavaScriptプログラム

```
let elem = document.getElementById("postage");
console.log("列数：" + elem.rows[0].cells.length);
```

← 列数を取得する

※「列数：4」が出力される


```
01  'use strict';
02  //Webページの読み込みが完了した時点で動作するユーザ定義関数
03  window.onload = function() {
04    let elem = document.getElementById("postage");
05    console.log("行数：" + elem.rows.length);
06    for(let row of elem.rows) {
07      console.log("列数：" + row.cells.length);
08    }
09    elem.rows[0].cells[0].innerHTML = "ご注文総額";
10  }
```

・実行結果

行数：5
列数：4 ←1行目
列数：4 ←2行目
列数：3 ←3行目
列数：2 ←4行目
列数：2 ←5行目

※Chromeの場合は開発ツール上で「F1キー」を押す→「Preferences」→「Console」の「Group similar messages in console」のチェックを外す

Webページの確認箇所 →

ご注文総額	本州・四国	北海道・九州	沖縄
3,000円以上	500円	700円	950円
5,000円以上	700円	850円	
10,000円以上	1,500円		
備考	購入金額が¥20,000円を超える場合は、 送料が変わる場合がありますのでお問合せください。		

第３章　JavaScript応用

3.2　HTMLタグ要素のプロパティ

3.2.5　styleプロパティ

・HTMLタグ要素のスタイル（CSSによる設定値、またはデフォルト値）
・styleオブジェクト（styleプロパティ）のプロパティ名に注意する
・styleオブジェクトの代表的なプロパティ

styleオブジェクトのプロパティ	CSSのプロパティ	対　象
color	color	色（文字色）
backgroundColor	background-color	背景色
fontsize	font-size	フォントサイズ
paddingXXX	padding-XXX	内側の余白
marginXXX	margin-XXX	外側の余白

※「XXX」部分には、Top（上）、Bottom（下）、Left（左）、Right（右）が入る

246

第３章　JavaScript応用

3.2　HTMLタグ要素のプロパティ

3.2.5　styleプロパティ

・paddingプロパティは、内側の余白を格納
【書式】Elementオブジェクト.style.paddingXXX

（例）

HTMLファイルの表示結果

赤枠　：境界線
緑枠　：内側の余白

JavaScriptプログラム

```
let elem = document.getElementById("inquiry");
elem.style.paddingLeft = "60px";
elem.style.paddingRight = "60px";
```

更新する

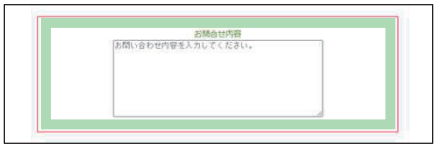
※Webページの段落「お問合せ内容」の内側の余白が更新される

247

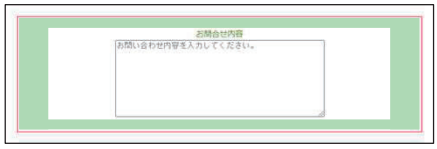
```
01  'use strict';
02  //Webページの読み込みが完了した時点で動作するユーザ定義関数
03  window.onload = function() {
04    let elem = document.getElementById("inquiry");
05    elem.style.paddingLeft = "60px";
06    elem.style.paddingRight = "60px";
07  }
```

・実行結果

［実行前］



［実行後］



赤枠：境界線，緑枠：内側の余白

3.2　HTMLタグ要素のプロパティ

3.2.5　styleプロパティ

・marginプロパティは、外側の余白を格納

【書式】Elementオブジェクト.style.marginXXX

（例）

HTMLファイルの表示結果

赤枠　　：境界線
緑枠　　：内側の余白
オレンジ枠：外側の余白

JavaScriptプログラム

```
let elem = document.getElementById("inquiry");
elem.style.marginLeft = "60px";
elem.style.marginRight = "60px";
```

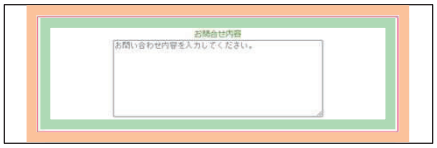
更新する

※Webページの段落「お問合せ内容」の外側の余白が更新される

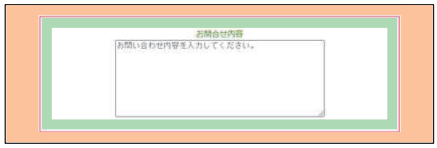
```
01  'use strict';
02  //Webページの読み込みが完了した時点で動作するユーザ定義関数
03  window.onload = function() {
04    let elem = document.getElementById("inquiry");
05    elem.style.marginLeft = "60px";
06    elem.style.marginRight = "60px";
07  }
```

・実行結果

［実行前］



［実行後］



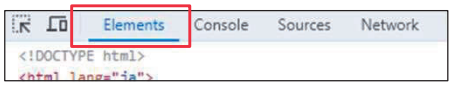
赤枠：境界線，緑枠：内側の余白，オレンジ枠：外側の余白

3.2　HTMLタグ要素のプロパティ

3.2.5　styleプロパティ

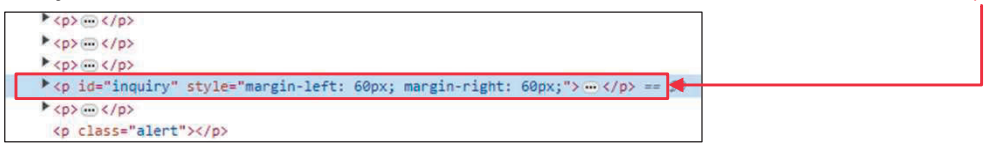
・HTMLタグ要素に適用されたスタイルを開発ツールで確認する方法

①開発者ツール上で「Elements」タブをクリックする



②bodyタグ以下を展開し、確認したいHTMLタグ要素を選択する

body → div → main → article → div → article → div → section → form → p



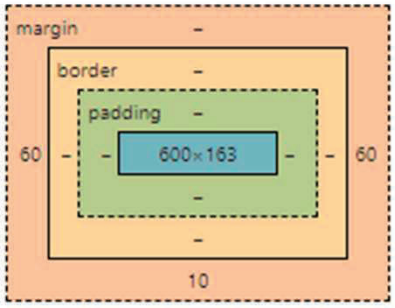
第３章　JavaScript応用

3.2　HTMLタグ要素のプロパティ

3.2.5　styleプロパティ

・HTMLタグ要素に適用されたスタイルを開発ツールで確認する方法

③右側の「Computed」タブを選択し、HTMLタグ要素に適用されたスタイルを図で確認する



（凡例）

青色　　：HTMLタグ要素のサイズ（px）
緑色　　：padding（px）
薄オレンジ色：境界線（px）
濃オレンジ色：margin（px）

252

第３章　JavaScript応用

3.3　HTMLタグ要素のメソッド

3.3.1　getAttributeメソッド

・Elementオブジェクトの属性値を返す

【書式】Elementオブジェクト.getAttribute(″属性名″)

（例）

HTMLファイル

```
<textarea name="otoiawase" rows="10" cols="50"
          placeholder="お問い合わせ内容を入力してください。">
</textarea>
```

JavaScriptプログラム

```
let elems = document.getElementsByName("otoiawase");
console.log(elems[0].getAttribute("rows"));
console.log(elems[0].getAttribute("cols"));
console.log(elems[0].getAttribute("placeholder"));
```

取得する

※「10」「50」「お問い合わせ内容を入力してください。」と出力される

253

```
01  'use strict';
02  //Webページの読み込みが完了した時点で動作するユーザ定義関数
03  window.onload = function() {
04    let elems = document.getElementsByName("otoiawase");
05    console.log(elems[0].getAttribute("rows"));
06    console.log(elems[0].getAttribute("cols"));
07    console.log(elems[0].getAttribute("placeholder"));
08  }
```

・実行結果

```
10                                  ← rows属性
50                                  ← cols属性
お問い合わせ内容を入力してください。  ← placeholder属性
```


3.3　HTMLタグ要素のメソッド

3.3.2　setAttributeメソッド

・Elementオブジェクトに属性名と属性値を設定（更新）する

【書式】Elementオブジェクト.setAttribute(″属性名″, ″属性値″)

（例）

HTMLファイル

```
<textarea name="otoiawase" rows="10" cols="50"
          placeholder="お問い合わせ内容を入力してください。">
</textarea>
```

JavaScriptプログラム

```
let elems = document.getElementsByName("otoiawase");
elems[0].setAttribute("rows", "20");
elems[0].setAttribute("cols", "80");
elems[0].setAttribute("placeholder", "内容は詳しくご入力ください。");
```

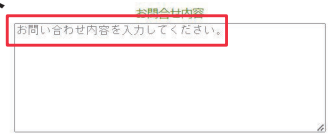
更新する

※Webページの「お問合せ内容」テキストエリアが更新される

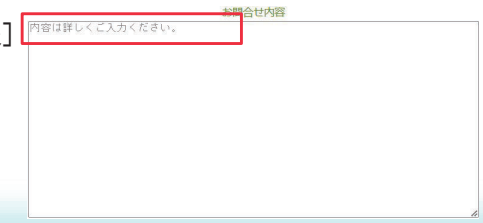
```
01  'use strict';
02  //Webページの読み込みが完了した時点で動作するユーザ定義関数
03  window.onload = function() {
04    let elems = document.getElementsByName("otoiawase");
05    elems[0].setAttribute("rows", "20");
06    elems[0].setAttribute("cols", "80");
07    elems[0].setAttribute("placeholder", "内容は詳しくご入力ください。");
08  }
```

・実行結果

［実行前］



［実行後］



3.3　HTMLタグ要素のメソッド

3.3.3　removeAttributeメソッド

・Elementオブジェクトから属性名と属性値を削除する

【書式】Elementオブジェクト.removeAttribute(″属性名″)

（例）

HTMLファイル

```
<textarea name="otoiawase" rows="10" cols="50"
          placeholder="お問い合わせ内容を入力してください。">
</textarea>
```

JavaScriptプログラム

```
let elems = document.getElementsByName("otoiawase");
elems[0].removeAttribute("placeholder");
```

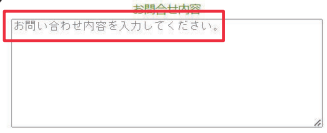
削除する

※Webページの「お問合せ内容」テキストエリアが更新される

```
01  'use strict';
02  //Webページの読み込みが完了した時点で動作するユーザ定義関数
03  window.onload = function() {
04    let elems = document.getElementsByName("otoiawase");
05    elems[0].removeAttribute("placeholder");
06  }
```

・実行結果

［実行前］



［実行後］

3.3　HTMLタグ要素のメソッド

3.3.4　hasAttributeメソッド

・Elementオブジェクトに属性名と属性値が存在するか（true／false）を確認する

【書式】Elementオブジェクト.hasAttribute(″属性名″)

（例）

HTMLファイル

```
<textarea name="otoiawase" rows="10" cols="50"
          placeholder="お問い合わせ内容を入力してください。">
</textarea>
```

JavaScriptプログラム

```
let elems = document.getElementsByName("otoiawase");
console.log(elems[0].hasAttribute("placeholder"));
elems[0].removeAttribute("placeholder");
console.log(elems[0].hasAttribute("placeholder"));
```

確認する

※「true」「false」と出力される

```
01  'use strict';
02  //Webページの読み込みが完了した時点で動作するユーザ定義関数
03  window.onload = function() {
04    let elems = document.getElementsByName("otoiawase");
05    console.log(elems[0].hasAttribute("placeholder"));
06    elems[0].removeAttribute("placeholder");          //placeholder属性を削除
07    console.log(elems[0].hasAttribute("placeholder"));
08  }
```

・**実行結果**

true　← placeholder属性の削除前
false　← placeholder属性の削除後

3.4　フォーム操作

3.4.1　値の設定と取得

・代表的な入力フォームと値

入力フォーム	概　要	プロパティ名	保持する値
text	テキストボックス	value	入力値（文字列）
textarea	テキストエリア	value	入力値（文字列）
checkbox	チェックボックス	checked value	選択状態（true／false） 選択値（文字列）
radio	ラジオボタン	checked value	選択状態（true／false） 選択値（文字列）
select	セレクトボックス	options[添字].selected options[添字].value selectedIndex	選択状態（true／false） 選択値（文字列） 選択番号（0以上の整数）

第３章　JavaScript応用

3.4　フォーム操作

3.4.1　値の設定と取得

・テキストボックス

【書式】Elementオブジェクト.value

（例１）

HTMLファイルの表示結果

名前：

JavaScriptプログラム

```
let elems = document.getElementsByName("name");
elems[0].value = "聖徳太子";
```

設定する

※Webページの「氏名」テキストボックスに「聖徳太子」と表示される

第３章　JavaScript応用

3.4　フォーム操作

3.4.1　値の設定と取得

（例２）

HTMLファイルの表示結果

名前：聖徳太子

JavaScriptプログラム

```
let elems = document.getElementsByName("name");
console.log(elems[0].value);
```

取得する

※フォームの入力値が出力される

```
01  'use strict';
02  window.onload = function() {
03    let elems = document.getElementsByName("name");
04    elems[0].value = "聖徳太子";
05  }
06  //フォームの入力値／選択値が送信されるときに動作するユーザ定義関数
07  function formFunction() {
08    let elems = document.getElementsByName("name");
09    console.log(elems[0].value);
10    return false;      //送信をキャンセルする
11  }
```

・実行結果（Webページの送信ボタンを押す）

［Webページ］ 名前：聖徳太子　　［Console］聖徳太子

3.4　フォーム操作

3.4.1　値の設定と取得

・テキストエリア

【書式】Elementオブジェクト.value

（例１）

HTMLファイルの表示結果

お問合せ内容

お問い合わせ内容を入力してください。

JavaScriptプログラム

```
let elems = document.getElementsByName("otoiawase");
elems[0].value = "庭に植物を植えようと思っています。…";
```

設定する

※Webページの「お問合せ内容」テキストエリアに「庭に植物を植えようと思っています。…」と表示される

3.4　フォーム操作

3.4.1　値の設定と取得

（例２）

HTMLファイル

お問合せ内容

庭に植物を植えようと思っています。
紫陽花、向日葵、金木犀で迷っています。
どれかを優先的に植えるとしたらどれが良いでしょうか？

JavaScriptプログラム

取得する

```
let elems = document.getElementsByName("otoiawase");
console.log(elems[0].value);
```

※フォームの入力値が出力される


```
01  'use strict';
02  window.onload = function() {
03    let elems = document.getElementsByName("otoiawase");
04    elems[0].value = "庭に植物を植えようと思っています。…";
05  }
06  //フォームの入力値／選択値が送信されるときに動作するユーザ定義関数
07  function formFunction() {
08    let elems = document.getElementsByName("otoiawase");
09    console.log(elems[0].value);
10    return false;      //送信をキャンセルする
11  }
```

・実行結果（Webページの送信ボタンを押す）

［Webページ］

お問合せ内容

庭に植物を植えようと思っています。
紫陽花、向日葵、金木犀で迷っています。
どれかを優先的に植えるとしたらどれが良いでしょうか？

［Console］庭に植物を植えようと思っています。…

第３章　JavaScript応用

3.4　フォーム操作

3.4.1　値の設定と取得

・チェックボックス

【書式】Elementオブジェクト.checked
Elementオブジェクト.value

（例１）

HTMLファイルの表示結果

好きな花の名前：□ あじさい □ ひまわり ☑ きんもくせい □ コスモス

JavaScriptプログラム

```
let elems = document.getElementsByName("flower");
elems[3].checked = true;
```

選択する

※Webページの「コスモス」チェックボックスが選択状態になる

第３章　JavaScript応用

3.4　フォーム操作

3.4.1　値の設定と取得

（例２）

HTMLファイル

好きな花の名前：☑ あじさい ☑ ひまわり □ きんもくせい ☑ コスモス

JavaScriptプログラム

```
let elems = document.getElementsByName(" flower ");
console.log(elems[3].value + "  (" + elems[3].checked + ")  ");
```

取得する

※入力フォームの選択値と選択状態が出力される

```
01  'use strict';
02  //フォームの入力値／選択値が送信されるときに動作するユーザ定義関数
03  function formFunction() {
04    let elems = document.getElementsByName("flower");
05    for(let elem of elems) {
06      console.log(elem.value + " (" + elem.checked + ") ");
07    }
08    return false;     //送信をキャンセルする
09  }
```

・実行結果（Webページのチェックボックスを選択して送信ボタンを押す）

［Webページ］ 好きな花の名前：☑ あじさい ☑ ひまわり ☐ きんもくせい ☑ コスモス

［Console］ あじさい（true）
ひまわり（true）
きんもくせい（false）
コスモス（true）

3.4　フォーム操作

3.4.1　値の設定と取得

・ラジオボタン

【書式】Elementオブジェクト.checked
Elementオブジェクト.value

（例１）

HTMLファイルの表示結果

性別：○ 男性 ◉ 女性

JavaScriptプログラム

```
let elems = document.getElementsByName("gender");
elems[0].checked = true;
```

選択する

※Webページの「性別」ラジオボタンの「男性」が選択状態になる

3.4　フォーム操作

3.4.1　値の設定と取得

（例２）

HTMLファイル

性別：◉ 男性 ○女性

JavaScriptプログラム

```
let elems = document.getElementsByName("gender");
console.log(elems[0].value + "  (" + elems[0].checked + ")  ");
```

取得する

※入力フォームの選択値と選択状態が出力される


```
01  'use strict';
02  //フォームの入力値／選択値が送信されるときに動作する
03  function formFunction() {
04    let elems = document.getElementsByName("gender");
05    for(let elem of elems) {
06      console.log(elem.value + "  (" + elem.checked + ")  ");
07    }
08    return false;     //送信をキャンセルする
09  }
```

・実行結果（Webページのラジオボタンを選択して送信ボタンを押す）

［Webページ］ 性別：◉ 男性 ○女性

［Console］ 男性 (true)

女性 (false)

第３章　JavaScript応用

3.4　フォーム操作

3.4.1　値の設定と取得

・セレクトボックス

【書式】Elementオブジェクト.options[添字].selected
Elementオブジェクト.options[添字].value
Elementオブジェクト.selectedIndex

（例１）

HTMLファイルの表示結果

好きな色：blue

JavaScriptプログラム

```
let elems = document.getElementsByName("color");
let elem = elems[0];
elem.options[2].selected = true;
```

選択する

※Webページの「好きな色」セレクトボックスの「yellow」が選択状態になる

第３章　JavaScript応用

3.4　フォーム操作

3.4.1　値の設定と取得

（例２）

HTMLファイル

好きな色：yellow

JavaScriptプログラム

```
let elems = document.getElementsByName("color");
let elem = elems[0];
let i = elem.selectedIndex;
console.log(elem.options[i].value + "  (" + elem.options[i].selected + ")  ");
```

取得する

※入力フォームの選択値と選択状態が出力される

```
01  'use strict';
02  //フォームの入力値／選択値が送信されるときに動作するユーザ定義関数
03  function formFunction() {
04    let elems = document.getElementsByName("color");
05    for(let opt of elems[0].options) {
06      console.log(opt.value + " (" + opt.selected + ") ");
07    }
08    return false;     //送信をキャンセルする
09  }
```

・実行結果（Webページの送信ボタンを押す）

［Webページ］ 好きな色：yellow

［Console］ 青 (false)
赤 (false)
黄 (true)
橙 (false)

3.4　フォームの操作

3.4.2　パスワードのチェック

・「パスワード」と「パスワード（確認）」の入力値を比較

（例）

HTMLファイル

```
<input type="password" name="password" placeholder="…" required>
<input type="password" name="password2" placeholder="…" required>
```

JavaScriptプログラム

```
let elems = document.getElementsByName("password");
let elems2 = document.getElementsByName("password2");
if(elems[0].value == elem2s[0].value) {
  console.log("パスワードが一致しました。");
} else {
  console.log("パスワードが一致しませんでした。");
}
```

```
01  'use strict';
02  //フォームの入力値／選択値が送信されるときに動作するユーザ定義関数
03  function formFunction() {
04     let elems = document.getElementsByName("password");
05     let elems2 = document.getElementsByName("password2");
06     if(elems[0].value == elems2[0].value) {
07        console.log("パスワードが一致しました。");
08     } else {
09        console.log("パスワードが一致しませんでした。");
10     }
11     return false;      //送信をキャンセルする
12  }
```

・実行結果（Webページの送信ボタンを押す）
［passとpassの場合］ パスワードが一致しました。
［passとwordの場合］ パスワードが一致しませんでした。

3.4　フォームの操作

3.4.3　正規表現によるチェック

・フォームの入力値を正規表現で比較
【書式】（正規表現）.test(フォームの入力値)

（例）

HTMLファイル

```
<input type="tel" name="telephone" placeholder="…">
```

JavaScriptプログラム

```
let elems = document.getElementsByName("telephone");
let regex = /^0¥d{1,4}-¥d{1,4}-¥d{3,4}$/;      //電話番号の正規表現
if(regex.test(elems[0].value)) {                //一般的に「== true」は省略
   console.log("電話番号は正しい書式です。");
} else {
   console.log("電話番号は誤った書式です。");
}
```

```
01  'use strict';
02  //フォームの入力値／選択値が送信されるときに動作するユーザ定義関数
03  function formFunction() {
04    let elems = document.getElementsByName("telephone");
05    let regex = /^0¥d{1,4}-¥d{1,4}-¥d{3,4}$/;
06    if(regex.test(elems[0].value)) {
07      console.log("電話番号は正しい書式です。");
08    } else {
09      console.log("電話番号は誤った書式です。");
10    }
11    return false;      //送信をキャンセルする
12  }
```

・実行結果（Webページの送信ボタンを押す）
［0120-999-888の場合］　電話番号は正しい書式です。
［0120-999-888aの場合］電話番号は誤った書式です。

3.4　フォームの操作

3.4.4　クラスの追加と削除

・HTMLタグ要素のクラス属性に設定されているクラス群に特定のクラス名を追加
【書式】Elementオブジェクト.classList.add(″クラス名″)

（例）

```
JavaScriptプログラム
let elems = document.getElementsByName("telephone");
elems[0].classList.add("error");        //クラス群にerrorクラスを追加
```

・HTMLタグ要素のクラス属性に設定されているクラス群から特定のクラス名を削除
【書式】Elementオブジェクト.classList.remove(″クラス名″)

（例）

```
JavaScriptプログラム
let elems = document.getElementsByName("telephone");
elems[0].classList.remove("error");     //クラス群からerrorクラスを削除
```

```
01  'use strict';
02  //フォームの入力値／選択値が送信されるときに動作するユーザ定義関数
03  function formFunction() {
04    let elems = document.getElementsByName("telephone");
05    let regex = /^0¥d{1,4}-¥d{1,4}-¥d{3,4}$/;
06    if(regex.test(elems[0].value)) {
07      elems[0].classList.remove("error");
08    } else {
09      elems[0].classList.add("error");
10    }
11    return false;      //送信をキャンセルする
12  }
```

・実行結果（Webページの送信ボタンを押す）

［正しい書式の場合］ 電話番号：0120-999-888

［誤った書式の場合］ 電話番号：0120-999-888a

282

JavaScript
第４章
イベント処理

【本章学習内容】

本章では、イベント処理について学習します。

283

第４章　イベント処理

４.１　イベントハンドラとイベントリスナ

４.１.１　イベントの種類

・イベントとは、Webブラウザ上で生じる変化のこと

［ウィンドウイベント］

- Webページが読み込まれた
- Webページが表示された
- Webブラウザのサイズが変わった　など

［フォームイベント］

- ラジオボタンやチェックボックスがクリックされた
- テキストボックスでキーが押された
- セレクトボックスの選択項目が変更された
- フォームの入力値／選択値が送信された　など

284

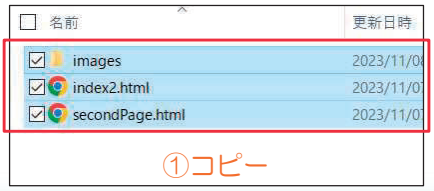
第４章　イベント処理

4.1　イベントハンドラとイベントリスナ

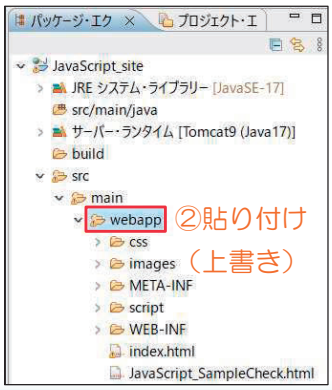
4.1.1　イベントの種類

- 第４章の配布プログラム「JavaScript第４章.zip」を展開（解凍）
- 「index2.html」ファイル、「secondPage.html」ファイル、「images」フォルダをコピー
- Eclipseの「webapp」フォルダ上で貼り付け

エクスプローラ



Eclipse



285

第４章　イベント処理

4.1　イベントハンドラとイベントリスナ

4.1.1　イベントの種類

- Eclipse上で「index2.html」を選択
- メニューの「実行」→「実行」→「サーバで実行」を選択
- Webブラウザに「第４章 イベント処理」が表示される

※右のWebページ内のHTMLタグ要素に対して、JavaScriptのプログラムから操作する方法を学習する

Webページ

286

第4章　イベント処理

4.1　イベントハンドラとイベントリスナ

4.1.1　イベントの種類

- 「script」フォルダに新規JavaScriptファイル「chap4.js」を作成
- 「index2.html」ファイルを開き、外部ファイル「script/chap4.js」を読み込むように修正
- 下記コードの9、10行目を追加

```
   :
06  <link href="css/reset.css" rel="stylesheet" type="text/css">
07  <link href="css/comon.css" rel="stylesheet" type="text/css">
08  <link href="css/main.css" rel="stylesheet" type="text/css">
09  <script src="script/chap4.js"></script>
10  <link rel="icon" href="data:,">
   :
```


第4章　イベント処理

4.1　イベントハンドラとイベントリスナ

4.1.2　イベントハンドラ

- イベントハンドラとは、イベントの発生（発火）時に呼び出される関数のこと
- 代表的なイベントとイベントハンドラの対応

イベント名	イベント概要	イベントハンドラ名
load	Webページが読み込まれた	onload
pageshow	Webページが表示された	onpageshow
resize	Webブラウザのサイズが変わった	onresize
click	マウスでクリックされた	onclick
keypress	キーボードでキーが押された	onkeypress
change	選択項目が変更された	onchange
submit	フォームの入力値／選択値が送信される	onsubmit

第４章　イベント処理

4.1　イベントハンドラとイベントリスナ

4.1.2　イベントハンドラ

・ウィンドウイベントのイベントハンドラを設定する

【書式】window.イベントハンドラ名 = function() {　処理　}

（例）
```
//Webページの読み込みが完了した時点で動作する
window.onload = function() {
  console.log("loadイベント発生");
}
```

・フォームイベントのイベントハンドラを設定する

【書式】入力フォームのタグに属性としてイベントハンドラ名と処理を記述する

（例）
```
<!--フォームの入力値／選択値が送信されるときに動作する-->
<form action="" name="form1" id="form" method="post"
      onsubmit="return formFunction();">
```

※予約語「return」は、onsubmitイベントハンドラの場合のみ記述する

289

第４章　イベント処理

```
01  'use strict';
02  //Webページの読み込みが完了した時点で動作する
03  window.onload = function() {
04    console.log("loadイベント発生");
05  }
06  //フォームの入力値／選択値が送信されるときに動作するユーザ定義関数
07  function formFunction() {
08    console.log("submitイベント発生");
09    return false;      //送信をキャンセルする
10  }
```

呼出し

HTMLファイル（入力済み）

```
58  <form action="" name="form1" id ="form" method="post" onsubmit="return formFunction();">
```

・実行結果

loadイベント発生　← Webページの読み込みが完了した時点

submitイベント発生　← 送信ボタンを押した時点

290

第４章　イベント処理

4.1　イベントハンドラとイベントリスナ

4.1.3　イベントリスナ

・イベントリスナとは、イベントの発生（発火）を監視し、イベントが発生（発火）した際に複数の関数を呼び出す仕組みのこと

・フォームイベントのイベントリスナを追加する

【書式】Elementオブジェクト.addEventListener(″イベント名″, 関数名)

（例）
```
window.onload = function() {
  let elems = document.getElementsByName("whatsNewButton");
  //お知らせ更新ボタンにイベントリスナを追加する
  elems[0].addEventListener("click", whatsNewButtonClick);
}

//お知らせ更新ボタンをクリックした時点で動作するユーザ定義関数
function whatsNewButtonClick() {
  console.log("clickイベント発生");
}
```


第４章　イベント処理

```
01  'use strict';
02  //Webページの読み込みが完了した時点で動作する
03  window.onload = function() {
04    let elems = document.getElementsByName("whatsNewButton");
05    //お知らせ更新ボタンにイベントリスナを追加する
06    elems[0].addEventListener("click", whatsNewButtonClick);
07  }
08  //お知らせ更新ボタンをクリックした時点で動作するユーザ定義関数
09  function whatsNewButtonClick() {
10    console.log("clickイベント発生");
11  }
```

呼出し

・実行結果

clickイベント発生　← お知らせ更新ボタンをクリックした時点

第４章　イベント処理

4.2　ウィンドウイベント処理

4.2.1　loadイベント

・Webページが読み込まれたときに発生（発火）するイベント
・Webブラウザの「再読み込みボタン」をクリックした場合も発生（発火）する
・Webブラウザにキャッシュされ たWebページを表示した場合は発生（発火）しない
・イベントハンドラで記述した場合

（書式）
```
window.onload = function() {
    処理
}
```

・イベントリスナで記述した場合

（書式）
```
window.addEventListener("load", 関数名);
```

293

第４章　イベント処理

```
01  'use strict';
02  //Webページの読み込みが完了した時点で動作する
03  window.onload = function() {
04      let date = new Date();                          //日付オブジェクトを生成
05      let ymd = date.toLocaleDateString("sv-SE");     //YYYY-MM-DD形式に変換
06      let elems = document.getElementsByName("date");
07      elems[0].value = ymd;
08  }
```

・**実行結果**

［実行前］ 日付：年 /月/日 🗓　　［実行後］ 日付：2023/11/09 🗓 ← **本日の日付**

294

第４章　イベント処理

4.2　ウィンドウイベント処理

4.2.2　pageshowイベント

- Webページが表示されたときに発生（発火）するイベント
- Webブラウザの「戻るボタン」「進むボタン」「再読み込みボタン」をクリックした場合も発生（発火）する
- Webブラウザにキャッシュされ たWebページを表示した場合も発生（発火）する
- イベントハンドラで記述した場合

  （書式）window.onpageshow = function() {
  　　処理
  　}
- イベントリスナで記述した場合

  （書式）window.addEventListener("pageshow", 関数名);

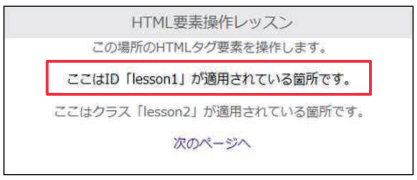
第４章　イベント処理

```
01  'use strict';
02  //Webページを表示した時点で動作する
03  window.onpageshow = function() {
04    let date = new Date();                //日付オブジェクトを生成
05    let year = date.getFullYear();        //年を取得
06    let month = date.getMonth() + 1;      //月（0～11）を取得
07    let day = date.getDate();             //日を取得
08    let hour = date.getHours();           //時を取得
09    let min = date.getMinutes();          //分を取得
10    let sec = date.getSeconds();          //秒を取得
11    let elem = document.getElementById("lesson1");
12    elem.innerHTML = year + "年" + month + "月" + day + "日" + hour + "時" + min + "分" + sec + "秒";
13  }
```

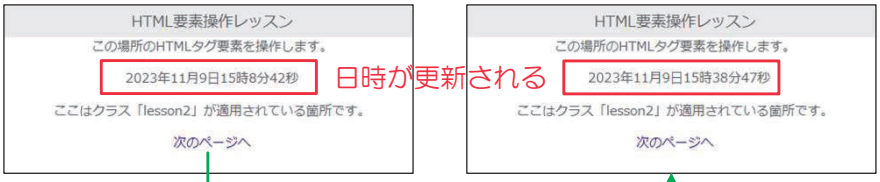
第４章　イベント処理

・実行結果

［実行前］



［実行後］



・リンクをクリックし、次のページが表示されたら「戻るボタン」を押す

・「再読み込みボタン」を押す

297

第４章　イベント処理

［再掲］

```
01  'use strict';
02  //Webページを表示した時点で動作する
03  window.onpageshow = function() {
04    let date = new Date();              //日付オブジェクトを生成
05    let year = date.getFullYear();      //年を取得
06    let month = date.getMonth() + 1;    //月（0～11）を取得
07    let day = date.getDate();           //日を取得
08    let hour = date.getHours();         //時を取得
09    let min = date.getMinutes();        //分を取得
10    let sec = date.getSeconds();        //秒を取得
11    let elem = document.getElementById("lesson1");
12    elem.innerHTML = year + "年" + month + "月" + day + "日" + hour + "時" + min + "分" + sec + "秒";
13  }
```

298

第４章　イベント処理

4.2　ウィンドウイベント処理

4.2.3　resizeイベント

・Webブラウザのサイズが変わったときに発生（発火）するイベント

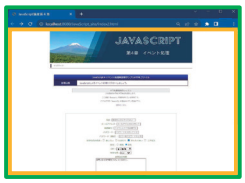
・イベントハンドラで記述した場合

（書式）window.onresize = function() {
　　処理
　}

・イベントリスナで記述した場合

（書式）window.addEventListener("resize", 関数名);

Webブラウザ



・ウィンドウ関連のサイズは、Windowオブジェクトのプロパティに格納されている

プロパティ名		概　要
outerWidth	outerHeight	Webブラウザの外周の幅と高さ（　　枠）
innerWidth	innerHeight	Webブラウザの表示領域の幅と高さ（　　枠）

299

第４章　イベント処理

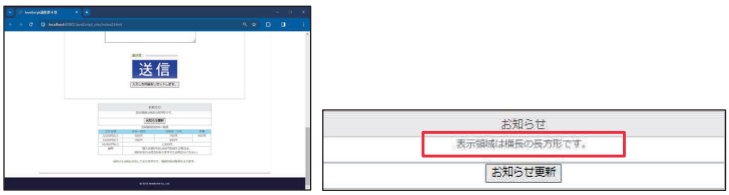
```
01  'use strict';
02  //Webブラウザのサイズを変更した時点で動作する
03  window.onresize = function() {
04    let width = window.innerWidth;       //Webブラウザの表示領域の幅
05    let height = window.innerHeight;     //Webブラウザの表示領域の高さ
06    let elems = document.getElementsByClassName("whatsNewMessage");
07    if(width > height) {
08      elems[0].innerHTML = "表示領域は横長の長方形です。";
09    } else if(width < height) {
10      elems[0].innerHTML = "表示領域は縦長の長方形です。";
11    } else {
12      elems[0].innerHTML = "表示領域は正方形です。";
13    }
14  }
```

300

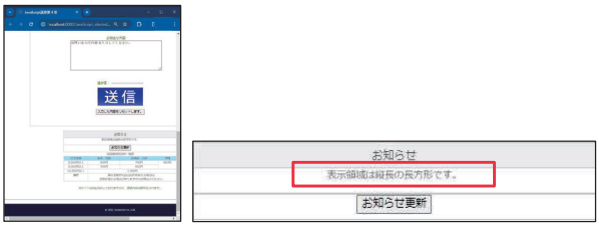
第４章　イベント処理

・実行結果

［Webブラウザが横長の場合］



［Webブラウザが縦長の場合］



301

第４章　イベント処理

［再掲］

```
01  'use strict';
02  //Webブラウザのサイズを変更した時点で動作する
03  window.onresize = function() {
04    let width = window.innerWidth;       //Webブラウザの表示領域の幅
05    let height = window.innerHeight;     //Webブラウザの表示領域の高さ
06    let elems = document.getElementsByClassName("whatsNewMessage");
07    if(width > height) {
08      elems[0].innerHTML = "表示領域は横長の長方形です。";
09    } else if(width < height) {
10      elems[0].innerHTML = "表示領域は縦長の長方形です。";
11    } else {
12      elems[0].innerHTML = "表示領域は正方形です。";
13    }
14  }
```

302

第４章　イベント処理

4.3　フォームイベント処理

4.3.1　clickイベント

・マウスでクリックされたときに発生（発火）するイベント

・イベントハンドラで記述した場合

（書式①）HTMLファイル

```
<input type="入力フォームの種類" … onclick="処理;">
<textarea … onclick="処理;"></textarea>　など
```

（書式②）JavaScriptプログラム

```
Elementオブジェクト.onclick = function() {
    処理
}
```

・イベントリスナで記述した場合

（書式③）JavaScriptプログラム

```
Elementオブジェクト.addEventListener("click", 関数名);
```

303

第４章　イベント処理

HTMLファイル

59行目のinputタグ

```
<input type="text" name="name" placeholder="名前を…" required
       onclick="this.classList.add('selectedElement');">
```

60行目のinputタグ

```
<input type="email" name="mail" placeholder="メールアドレスを…" required
       onclick="this.classList.add('selectedElement');">
```

61行目のinputタグ

```
<input type="tel" name="telephone" placeholder="ハイフン入りで電話番号を…" required
       onclick="this.classList.add('selectedElement');">
```

62行目のinputタグ

```
<input type="password" name="password" placeholder="パスワードを…" required
       onclick="this.classList.add('selectedElement');">
```

63行目のinputタグ

```
<input type="password" name="password2" placeholder="もう一度パスワードを…" required
       onclick="this.classList.add('selectedElement');">
```

※予約語thisは、関数の呼び出し元のオブジェクト（Elementオブジェクト）を示す

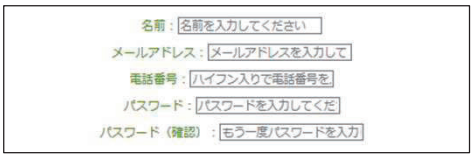
HTMLタグ要素内のJavaScriptプログラムではシングルクォート記号を使用する

304

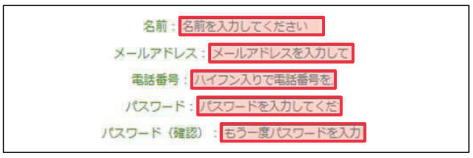
第４章　イベント処理

・実行結果

［クリックする前］



［クリックした後］



305

第４章　イベント処理

HTMLファイル　［再掲］

59行目のinputタグ
```
<input type="text" name="name" placeholder="名前を…" required
       onclick="this.classList.add('selectedElement');">
```
60行目のinputタグ
```
<input type="email" name="mail" placeholder="メールアドレスを…" required
       onclick="this.classList.add('selectedElement');">
```
61行目のinputタグ
```
<input type="tel" name="telephone" placeholder="ハイフン入りで電話番号を…" required
       onclick="this.classList.add('selectedElement');">
```
62行目のinputタグ
```
<input type="password" name="password" placeholder="パスワードを…" required
       onclick="this.classList.add('selectedElement');">
```
63行目のinputタグ
```
<input type="password" name="password2" placeholder="もう一度パスワードを…" required
       onclick="this.classList.add('selectedElement');">
```

※予約語thisは、関数の呼び出し元のオブジェクト（Elementオブジェクト）を示す
HTMLタグ要素内のJavaScriptプログラムではシングルクォート記号を使用する

306

第４章　イベント処理

4.3　フォームイベント処理

4.3.2　keydownイベント

・キーボードでキーが押されたときに発生（発火）するイベント

・イベントハンドラで記述した場合

（書式①）HTMLファイル

```
<input type="入力フォームの種類" … onkeydown="処理;">
<textarea … onkeydown="処理;"></textarea>　など
```

（書式②）JavaScriptプログラム

```
Elementオブジェクト.onkeydown = function() {
    処理
}
```

・イベントリスナで記述した場合

（書式③）JavaScriptプログラム

```
Elementオブジェクト.addEventListener("keydown", 関数名);
```


第４章　イベント処理

HTMLファイル

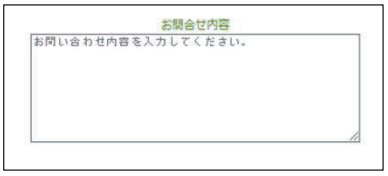
75行目のtextareaタグ

```
<textarea name="otoiawase" rows="10" cols="50" placeholder="お問い合わせ内容を…"
          onkeydown="this.classList.add('selectedElement');">
</textarea>
```

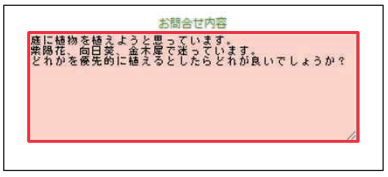
※予約語thisは、関数の呼び出し元のオブジェクト（Elementオブジェクト）を示す
HTMLタグ要素内のJavaScriptプログラムではシングルクォート記号を使用する

・**実行結果**

［キーを押す前］



［キーを押した後］



第４章　イベント処理

4.3　フォームイベント処理

4.3.3　changeイベント

・選択項目が変更されたときに発生（発火）するイベント

・イベントハンドラで記述した場合

（書式①）HTMLファイル

```
<input type="入力フォームの種類" … onchange="処理;">
<textarea … onchange="処理;"></textarea>
<select … onchange="処理;"></select>　など
```

（書式②）JavaScriptプログラム

```
Elementオブジェクト.onchange = function() {
   処理
}
```

・イベントリスナで記述した場合

（書式③）JavaScriptプログラム

```
Elementオブジェクト.addEventListener("change", 関数名);
```

309

第４章　イベント処理

HTMLファイル

67～72行目のselectタグ

```
<select name="color" onchange="changeFunction(this);">
  <option value="青">blue</option>
  <option value="赤">red</option>
  <option value="黄">yellow</option>
  <option value="橙">orange</option>
</select>
```

JavaScriptプログラム

```
01  'use strict';
02  //選択項目が変更された時点で動作するユーザ定義関数
03  function changeFunction(elem) {
04    let elem2 = document.getElementsByName("otoiawase");
05    let colorName = elem.options[elem.selectedIndex].textContent;     //選択表示名を取得
06    elem2[0].style.backgroundColor = colorName;                       //背景色を変更
07  }
```

※予約語thisは、関数の呼び出し元のオブジェクト（Elementオブジェクト）を示す

310

第４章　イベント処理

・実行結果

［選択項目を変更する前］

［選択項目を「yellow」に変更した後］

311

第４章　イベント処理

HTMLファイル　［再掲］

67～72行目のselectタグ

```
<select name="color" onchange="changeFunction(this);">
  <option value="青">blue</option>
  <option value="赤">red</option>
  <option value="黄">yellow</option>
  <option value="橙">orange</option>
</select>
```

JavaScriptプログラム

```
01  'use strict';
02  //選択項目が変更された時点で動作するユーザ定義関数
03  function changeFunction(elem) {
04    let elem2 = document.getElementsByName("otoiawase");
05    let colorName = elem.options[elem.selectedIndex].textContent;     //選択表示名を取得
06    elem2[0].style.backgroundColor = colorName;                       //背景色を変更
07  }
```

※予約語thisは、関数の呼び出し元のオブジェクト（Elementオブジェクト）を示す

312

第４章　イベント処理

4.3　フォームイベント処理

4.3.4　submitイベント

・フォームの入力値／選択値が送信されるときに発生（発火）するイベント

・イベントハンドラで記述した場合

（書式①）HTMLファイル

<fom action="送信先URL" ･･･ onsubmit="return 関数名();">

（書式②）JavaScriptプログラム

formタグのElementオブジェクト.onsubmit = function() {

　処理

}

・イベントリスナで記述した場合

（書式③）JavaScriptプログラム

formタグのElementオブジェクト.addEventListener("submit", 関数名);

313

第４章　イベント処理

HTMLファイル（入力済み）

58行目のformタグ

```
<form action="" name="form1" id="form" method="post" onsubmit="return formFunction();">
```

JavaScriptプログラム

```
01  'use strict';
02  //フォームの入力値／選択値が送信されるときに動作するユーザ定義関数
03  function formFunction() {
04    let elems = document.getElementsByName("password");
05    let elems2 = document.getElementsByName("password2");
06    if(elems[0].value == elems2[0].value) {
07      return true;      //送信する
08    } else {            //↓ビルトイン関数でダイアログを表示
09      alert("パスワードが一致しないため、送信をキャンセルします。");
10      return false;     //送信をキャンセルする
11    }
12  }
```

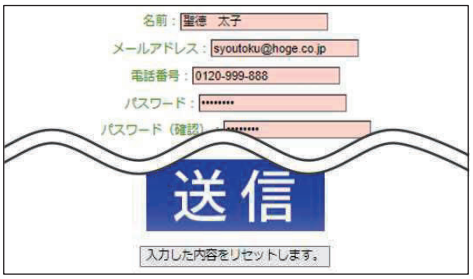
314

第４章　イベント処理

・実行結果

［送信ボタンを押す前］

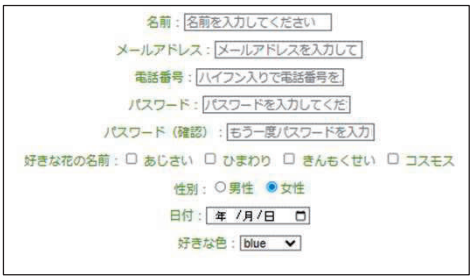
（事前に以下の設定を行ってください）



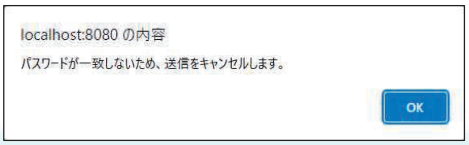
※Chromeの場合は
「Google Chromeの設定」→「設定」→「プライバシーとセキュリティ」→「セキュリティ」→「標準保護機能」→「データ侵害によりパスワードが漏洩した場合に警告する」のチェックを外す

［送信ボタンを押した後］

（パスワードが一致した場合）



（パスワードが一致しなかった場合）



315

第４章　イベント処理

HTMLファイル（入力済み）　［再掲］

58行目のformタグ

```
<form action="" name="form1" id="form" method="post" onsubmit="return formFunction();">
```

JavaScriptプログラム

```
01  'use strict';
02  //フォームの入力値／選択値が送信されるときに動作するユーザ定義関数
03  function formFunction() {
04    let elems = document.getElementsByName("password");
05    let elems2 = document.getElementsByName("password2");
06    if(elems[0].value == elems2[0].value) {     //パスワードのチェック
07      return true;      //送信する
08    } else {                                    //ビルトイン関数でダイアログを表示
09      alert("パスワードが一致しないため、送信をキャンセルします。");
10      return false;     //送信をキャンセルする
11    }
12  }
```

316

第１章　プログラミング演習

事前準備

①Eclipse上で動的Webプロジェクトを作成する
　　プロジェクト名：chap1_exercise
②HTMLファイルを作成する
　　保存先　　：webapp
　　HTMLファイル名：chap1_exercise.html
③JavaScriptファイル保存用フォルダを作成する
　　保存先　　：webapp
　　フォルダ名　　：script

次の**問題１**～**問題５**を読んで、提示された**[要件]**と**[実行結果]**を満たすプログラムを作成しなさい。

問題１（難易度 ☆）
・あるスマートフォンショップの在庫管理を行うプログラムを作成する
・JavaScriptプログラムは、「script」フォルダの「exercise1.js」ファイルに保存する
・プログラム作成後に「chap1_exercise.html」ファイルを実行し、Webブラウザのコンソールに表示される実行結果を確認する

[要件]
・一次元配列の添字は対応している
・機種ごとに、機種名・在庫数・在庫総額を出力する
・全機種の在庫総額を出力する
・機種ごとに、在庫数が15台になるように機種名と発注台数を出力する
・全機種の発注総額を出力する

[実行結果]
xPhone23の在庫は12台です。
在庫総額は1,440,000円です。
　：

全機種の在庫総額は10,398,000円です。

xPhone23は3台発注が必要です。
発注金額は360,000円です。

spaceS21は発注不要です。
　：

発注総額は1,450,000円必要です。

JavaScript

exercise1.js

```
'use strict';
const names = ["xPhone23", "spaceS21", "XORPhone9", "dot8Pro", "WaterSense6", "Yesthing3"];
const prices = [120000, 90000, 85000, 115000, 38000, 95000];
const stocks = [12, 26, 10, 30, 41, 8];

//機種ごとに機種名・在庫数・在庫総額を出力する
let totalAmount = 0;                    //機種の在庫総額
let totalInventory = 0;                 //全機種の在庫総額
for(          ; i <           ;      ) {
  console.log(names[i] + "の在庫は" + stocks[i] + "台です。");
  totalAmount = prices[i] *       ;
  console.log("在庫総額は" +                        ("      ") + "円です。");
  totalInventory = totalInventory +          ;
  console.log("");                      //空行の出力
}

//全機種の在庫総額を出力する
console.log("全機種の在庫総額は" +                        ("      ") + "円です。");
console.log("");

//在庫数が15台になるように、機種名と発注台数を出力する
totalAmount = 0;
totalInventory = 0;
for(          ; j <           ;      ) {
  if(stocks[j] < 15) {
    console.log(names[j] + "は" +              + "台発注が必要です。");
    totalAmount = prices[j] *            ;
    totalInventory = totalInventory + totalAmount;
    console.log("発注金額は" +                        ("      ") + "円です。");
    console.log("");
  } else {
    console.log(names[j] + "は発注不要です。");
    console.log("");
  }
}

//全機種の発注総額を出力する
console.log("発注総額は" + totalInventory.toLocaleString("ja-JP") + "円必要です。");
```

chap1_exercise.html

※以下のようにコードを**修正**する

```
<head>
<script src="script/exercise1.js"></script>
<link rel="icon" href="data:,">
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
```

問題２（難易度 ☆☆）

・2000年～2100年の「うるう年」を求めるプログラムを作成する
・JavaScriptプログラムは、「script」フォルダの「exercise2.js」ファイルに保存する
・プログラム作成後に「chap1_exercise.html」ファイルを実行し、Webブラウザのコンソールに表示される実行結果を確認する

［要件］

・2000年から2100年までのうるう年を、うるう年配列に格納する
・うるう年の判定
　- **年が400で割り切れる　⇒　うるう年である**
　- **年が4で割り切れる、かつ、年が100で割り切れない　⇒　うるう年である**
　- 上記以外　⇒　うるう年でない
・うるう年配列に格納された「うるう年」を出力する
・プログラム実行日より過去のうるう年は【過去】を出力する

［実行結果］

```
1回目のうるう年は2000年でした。【過去】
2回目のうるう年は2004年でした。【過去】
3回目のうるう年は2008年でした。【過去】
4回目のうるう年は2012年でした。【過去】
5回目のうるう年は2016年でした。【過去】
6回目のうるう年は2020年でした。【過去】
7回目のうるう年は2024年です。
8回目のうるう年は2028年です。
9回目のうるう年は2032年です。
10回目のうるう年は2036年です。
11回目のうるう年は2040年です。
12回目のうるう年は2044年です。
13回目のうるう年は2048年です。
14回目のうるう年は2052年です。
15回目のうるう年は2056年です。
16回目のうるう年は2060年です。
17回目のうるう年は2064年です。
18回目のうるう年は2068年です。
19回目のうるう年は2072年です。
20回目のうるう年は2076年です。
21回目のうるう年は2080年です。
22回目のうるう年は2084年です。
23回目のうるう年は2088年です。
24回目のうるう年は2092年です。
25回目のうるう年は2096年です。
```

JavaScript

exercise2.js

```
01 'use strict';
02 const leapYear = [];          //うるう年配列
03
04 //2000年から2100年までのうるう年を、うるう年配列に格納する
05 let year = 2000;              //調査開始年（2000年）
06 let yearEnd = 2100;           //調査終了年（2100年）
07 for(let i = year; i <= yearEnd; i++) {
08
09
10
11
12
13 }
14
15 //うるう年配列に格納された「うるう年」を出力する
16 //プログラム実行日より過去のうるう年は【過去】を出力する
17 let dateObj = new Date();
18 let nowYear = dateObj.getFullYear();
19 for(let j = 0; j < leapYear.length; j++) {
20
21
22
23
24
25 }
```

chap1_exercise.html

※以下のようにコードを**修正**する

```
03 <head>
04 <script src="script/exercise2.js"></script>
05 <link rel="icon" href="data:,">
06 <meta charset="UTF-8">
07 <title>Insert title here</title>
08 </head>
```

問題３（難易度 ☆☆☆）

・あるPC修理店の総売上金額を求めるプログラムを作成する
・JavaScriptプログラムは、「script」フォルダの「exercise3.js」ファイルに保存する
・プログラム作成後に「chap1_exercise.html」ファイルを実行し、Webブラウザのコンソールに表示される実行結果を確認する

［要件］

・一次元配列は添字で対応している
・ユーザ定義関数を使用して総売上金額を求める
・総売上金額を出力する
・修理内容コード、修理台数、修理対応人数から求めた修理金額を返すユーザ定義関数を定義する
・修理内容コードとPC１台の修理費用は次のとおりである

修理内容コード	修理費用
1	2,000円
2	3,500円
3	5,800円
4	9,800円
5	15,000円

［実行結果］

総売上金額は286,000円です。

JavaScript

exercise3.js

```
01 'use strict';
02 const rCode = [1,3,5,4,2,3,3,2,4,5];          //修理内容コード
03 const rUnit = [2,3,1,2,3,4,2,3,1,2];          //修理台数
04 const rPeople = [1,2,1,2,2,3,1,2,1,2];        //修理対応人数
05
06 //ユーザ定義関数を使用して総売上金額を求める
07 let totalFee = 0;
08
09
10
11
12 //総売上金額を出力する
13 console.log("総売上金額は" + totalFee.toLocaleString("ja-JP") + "円です。")
14
15 //修理内容コード、修理台数、修理対応人数から求めた修理金額を返すユーザ定義関数
16 function repairFee(code, unit, people) {
17   let price = 0;
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34   return price * unit * people;
35 }
```

chap1_exercise.html

※以下のようにコードを**修正**する

```
03 <head>
04 <script src="script/exercise3.js"></script>
05 <link rel="icon" href="data:,">
06 <meta charset="UTF-8">
07 <title>Insert title here</title>
08 </head>
```

問題４（難易度 ☆☆☆☆）

・あるセンサーが出力したデータを分析するプログラムを作成する
・JavaScriptプログラムは、「script」フォルダの「exercise4.js」ファイルに保存する
・プログラム作成後に「chap1_exercise.html」ファイルを実行し、Webブラウザのコンソールに表示される実行結果を確認する

［要件］
・センサーの出力データを集計する
　［出力データの例］（CSV形式の文字列）
　　”a,b,e,c,e,a,c,b,e,c,a,b,b,a,c,b,b,b,c,a”
　［出力データの記号］
　　a：レベルａを示す
　　b：レベルｂを示す
　　c：レベルｃを示す
　　e：エラーコードを示す
・エラーコードの出力回数に合わせてメッセージと出力回数を出力する
　［出力様式］
　　- エラーコードの出力回数が０回の場合　　：
　　　メッセージ「センサーの出力データは優良です。」
　　　レベルａ～ｃとエラーコードの出力回数を表示する
　　- エラーコードの出力回数が４回以下の場合：
　　　メッセージ「センサーの出力データは許容範囲内です。」
　　　レベルａ～ｃとエラーコードの出力回数を表示する
　　- エラーコードの出力回数が５回以上の場合：
　　　メッセージ「センサーの出力データは許容範囲外のため破棄してください。」
　　　エラーコードの出力回数を表示する

［実行結果］
センサーの出力データは許容範囲内です。
aの出力回数：5回
bの出力回数：7回
cの出力回数：5回
eの出力回数：3回

exercise4.js

```
01  'use strict';
02  let sensorData = "a,b,e,c,e,a,c,b,e,c,a,b,b,a,c,b,b,b,c,a";
      :
```

chap1_exercise.html

※以下のようにコードを**修正**する

```
03  <head>
04  <script src="script/exercise4.js"></script>
05  <link rel="icon" href="data:,">
06  <meta charset="UTF-8">
07  <title>Insert title here</title>
08  </head>
```

問題５（難易度 ☆☆☆☆☆）

・国名の誤り部分を修正するプログラムを作成する
・JavaScriptプログラムは、「script」フォルダの「exercise5.js」ファイルに保存する
・プログラム作成後に「chap1_exercise.html」ファイルを実行し、Webブラウザのコンソールに表示される実行結果を確認する

［要件］

・国名の誤り部分を正規表現による置換によって修正する
　［国名の例］（二次元配列）
　　[[Chiiina", "Uniiited Kingdom"],
　　["Geeermany", "Beeelgium", "Switzeeerland"],
　　["United Staaates", "Japaaan", "Fraaance", "Portugaaal"]]
　［修正方法］
　　１行目：「iii」を「i」に置換する
　　２行目：「eee」を「e」に置換する
　　３行目：「aaa」を「a」に置換する
・修正後の国名を出力する

［実行結果］

```
1行目
  China
  United Kingdom,
2行目
  Germany
  Belgium
  Switzerland
3行目
  United States
  Japan
  France
  Portugal
```

exercise5.js

```
01  'use strict';
02  const countries = [["Chiiina", "Uniiited Kingdom"],
                       ["Geeermany", "Beeelgium", "Switzeeerland"],
                       ["United Staaates", "Japaaan", "Fraaance", "Portugaaal"]];
      :
```

chap1_exercise.html

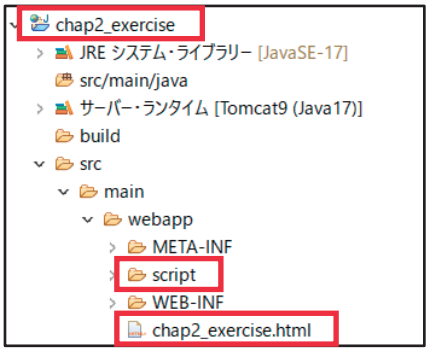
※以下のようにコードを**修正**する

```
03  <head>
04  <script src="script/exercise5.js"></script>
05  <link rel="icon" href="data:,">
06  <meta charset="UTF-8">
07  <title>Insert title here</title>
08  </head>
```

第２章　プログラミング演習

事前準備

①Eclipse上で動的Webプロジェクトを作成する
　　プロジェクト名：chap2_exercise
②HTMLファイルを作成する
　　保存先　　：webapp
　　HTMLファイル名：chap2_exercise.html
③JavaScriptファイル保存用フォルダを作成する
　　保存先　　：webapp
　　フォルダ名　　：script



次の**問題１**～**問題５**を読んで、提示された**[要件]**と**[実行結果]**を満たすプログラムを作成しなさい。

問題１（難易度 ☆）
・ある販売店の売上集計を行うプログラムを作成する
・JavaScriptプログラムは、「script」フォルダの「exercise1.js」ファイルに保存する
・プログラム作成後に「chap2_exercise.html」ファイルを実行し、Webブラウザのコンソールに表示される実行結果を確認する

[要件]
・一次元配列の添字は対応している
・売上データを集計し、全製品の売上額合計、売上最高額と売上最低額の添字と売上額、全製品の売上額平均を求める
・全製品の売上額合計と売上額平均、型番と売上最高額、型番と売上最低額を出力する

[実行結果]
全商品の売上額合計は1,267,000円です。
全商品の売上額平均は211,166円です。
売上最高額の型番はB03で、売上額は406,000円です。
売上最低額の型番はD30で、売上額は81,000円です。

JavaScript

exercise1.js

```
01  'use strict';
02  const typeNo = ["A01", "B03", "A90", "D30", "F89", "C40"];        //型番
03  const price = [1000, 1400, 2000, 2700, 3000, 4000];               //単価
04  const unit = [100, 290, 120, 30, 80, 50];                         //販売台数
05  let sumSales = 0;              //全製品の売上額合計
06  let avgSales = 0;              //全製品の売上額平均
07  let maxSales;                  //売上最高額の製品の売上額
08  let max = 0;                   //売上最高額の製品の添字
09  let minSales;                  //売上最低額の製品の売上額
10  let min = 0;                   //売上最低額の製品の添字
11
12  //売上データを集計し、全製品の売上額合計、売上最高額と売上最低額の添字と売上額、
13  //全製品の売上額平均を求める
14  maxSales = price[max] * unit[max];
15  minSales = price[min] * unit[min];
16  let sales = [                ];
17  sumSales = sales;
18  for(let i = 1; i < typeNo.length; i++) {
19    sales = price[i] * unit[i];
20    sumSales = [                ];
21    if(maxSales < sales) {
22      maxSales = [      ];
23      [    ] = i;
24    }
25    if(minSales > sales) {
26      minSales = [      ];
27      [    ] = i;
28    }
29  }
30  avgSales = Math.floor([                        ]);
31
32  //全製品の売上額合計と売上額平均、型番と売上最高額、型番と売上最低額を出力する
33  console.log("全製品の売上額合計は" + sumSales.toLocaleString("ja-JP") + "円です。");
34  console.log("全製品の売上額平均は" + avgSales.toLocaleString("ja-JP") + "円です。");
35  console.log("売上最高額の型番は" + [          ] + "で、売上額は"
36              + maxSales.toLocaleString("ja-JP") + "円です。");
37  console.log("売上最低額の型番は" + [          ] + "で、売上額は"
38              + minSales.toLocaleString("ja-JP") + "円です。");
```

chap2_exercise.html

※以下のようにコードを**修正**する

```
03  <head>
04  <script src="script/exercise1.js"></script>
05  <link rel="icon" href="data:,">
06  <meta charset="UTF-8">
07  <title>Insert title here</title>
08  </head>
```

問題２（難易度 ☆☆）

・受験者の合否判定を行うプログラムを作成する
・JavaScriptプログラムは、「script」フォルダの「exercise2.js」ファイルに保存する
・プログラム作成後に「chap2_exercise.html」ファイルを実行し、Webブラウザのコンソールに表示される実行結果を確認する

［要件］
・合格者の受験番号一覧から、任意の受験番号を探索する（二分探索法）
・探索後の結果により、合否判定を出力する
・合否判定
　- **合格者の受験番号一覧に、任意の受験番号が見つかった　⇒　合格**
　- **合格者の受験番号一覧に、任意の受験番号が見つからなかった　⇒　不合格**

［実行結果］
受験番号333は合格です。
受験番号439は不合格です。
受験番号889は合格です。

JavaScript

exercise2.js

```
'use strict';
const passNo = [235, 326, 333, 395, 438, 599, 656, 889, 891];     //合格者の受験番号一覧
const examNo = [333, 439, 889];                                    //任意の受験番号一覧

for(let i = 0; i < examNo.length; i++) {
  //合格者の受験番号一覧から、任意の受験番号を探索する（二分探索法）
  let le = 0;
  let ri = passNo.length - 1;
  let mi = Math.floor((le + ri) / 2);
  
  
  
  
  
  
  
  
  
  //探索後の結果により、合否判定を出力する
  
  
  
  
  
}
```

chap2_exercise.html

※以下のようにコードを**修正**する

```
<head>
<script src="script/exercise2.js"></script>
<link rel="icon" href="data:,">
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
```

問題３（難易度 ☆☆☆）

・あるスマートフォンショップの商品データを出力するプログラムを作成する
・JavaScriptプログラムは、「script」フォルダの「exercise3.js」ファイルに保存する
・プログラム作成後に「chap2_exercise.html」ファイルを実行し、Webブラウザのコンソールに表示される実行結果を確認する

［要件］
・一次元配列の添字は対応している
・在庫台数の降順に機種名、単価、在庫台数を整列する（基本選択法）
・整列後の機種名、単価、在庫台数を出力する

［実行結果］
機種名:WaterSense6　単価：WaterSense6　在庫台数：41
機種名:dot8Pro　単価：dot8Pro　在庫台数：30
機種名:spaceS21　単価：spaceS21　在庫台数：26
機種名:xPhone23　単価：xPhone23　在庫台数：12
機種名:XORPhone9　単価：XORPhone9　在庫台数：10
機種名:Yesthing3　単価：Yesthing3　在庫台数：8

JavaScript

exercise3.js

```
01 'use strict';
02 const names = ["xPhone23", "spaceS21", "XORPhone9",
03                "dot8Pro", "WaterSense6", "Yesthing3"];          //機種名
04 const prices = [120000, 90000, 85000, 115000, 38000, 95000];    //単価
05 const stocks = [12, 26, 10, 30, 41, 8];                        //在庫台数
06
07 //在庫台数の降順に機種名、単価、在庫台数を整列する（基本選択法）
08 for(let i = 0; i < names.length - 1; i++) {
09
10
11
12
13
14
15   exchange(names, i, k);
16   exchange(prices, i, k);
17   exchange(stocks, i, k);
18 }
19
20 //整列後の機種名、単価、在庫台数を出力する
21 for(let i = 0; i < names.length; i++) {
22   console.log("機種名:" + names[i] + "  "
23               + "単価：" + prices[i].toLocaleString("ja-JP") + "  "
24               + "在庫台数：" + stocks[i].toLocaleString("ja-JP"));
25 }
26
27 //交換処理を行うユーザ定義関数
28 function exchange(arr, x, y) {
29
30
31
32
33 }
```

chap2_exercise.html

※以下のようにコードを**修正**する

```
03 <head>
04 <script src="script/exercise3.js"></script>
05 <link rel="icon" href="data:,">
06 <meta charset="UTF-8">
07 <title>Insert title here</title>
08 </head>
```

問題４（難易度 ☆☆☆☆）

・ある試験の上位10件の得点を出力するプログラムを作成する
・JavaScriptプログラムは、「script」フォルダの「exercise4.js」ファイルに保存する
・プログラム作成後に「chap2_exercise.html」ファイルを実行し、Webブラウザのコンソールに表示される実行結果を確認する

［要件］
・試験の得点を降順に整列する（基本交換法）
・試験の上位10件の得点について、順位と得点を出力する

［実行結果］
上位10件：
第1位の得点は99点です。
第2位の得点は95点です。
第3位の得点は83点です。
第4位の得点は77点です。
第5位の得点は76点です。
第6位の得点は74点です。
第7位の得点は73点です。
第8位の得点は68点です。
第9位の得点は64点です。
第10位の得点は56点です。

exercise4.js

```
01  'use strict';
02  const scores = [95, 43, 7, 64, 56, 99, 68, 77, 4,18, 76, 6, 73, 74, 30, 31, 83, 33, 32, 54];
      :
```

chap2_exercise.html

※以下のようにコードを**修正**する

```
03  <head>
04  <script src="script/exercise4.js"></script>
05  <link rel="icon" href="data:,">
06  <meta charset="UTF-8">
07  <title>Insert title here</title>
08  </head>
```

問題５（難易度 ☆☆☆☆☆）

・ある試験の合格者の受験番号を昇順に出力するプログラムを作成する
・JavaScriptプログラムは、「script」フォルダの「exercise5.js」ファイルに保存する
・プログラム作成後に「chap2_exercise.html」ファイルを実行し、Webブラウザのコンソールに表示される実行結果を確認する

［要件］
・試験の合格者の受験番号を昇順に整列する（クイックソート）
・整列後の、試験の合格者の受験番号を出力する

［実行結果］
合格者の受験番号：
235
315
326
333
395
438
599
656
889
891

exercise5.js

```
01  'use strict';
02  const passNo = [438, 889, 333, 315, 599, 891, 326, 235, 395, 656];
      :
```

chap2_exercise.html

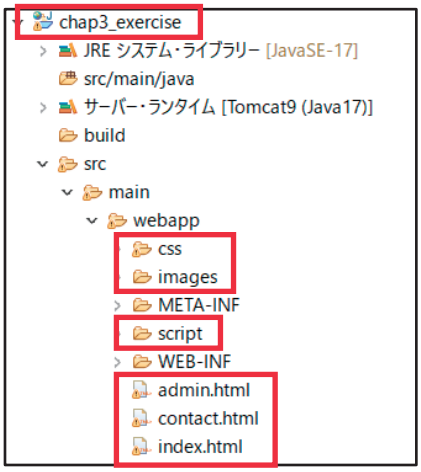
※以下のようにコードを**修正**する

```
03  <head>
04  <script src="script/exercise5.js"></script>
05  <link rel="icon" href="data:,">
06  <meta charset="UTF-8">
07  <title>Insert title here</title>
08  </head>
```

第３章　プログラミング演習

事前準備

①Eclipse上で動的Webプロジェクトを作成する
　プロジェクト名：chap3_exercise
②配布データをコピーする
　配布フォルダ名：css
　　images
　配布ファイル名：admin.html
　　contact.html
　　index.html
③配布データを貼り付ける
　貼付先　　：webapp
④JavaScriptファイル保存用フォルダを作成する
　保存先　　：webapp
　フォルダ名　：script



次の**問題１**～**問題５**を読んで、提示された**[要件]**と**[実行結果]**を満たすプログラムを作成しなさい。

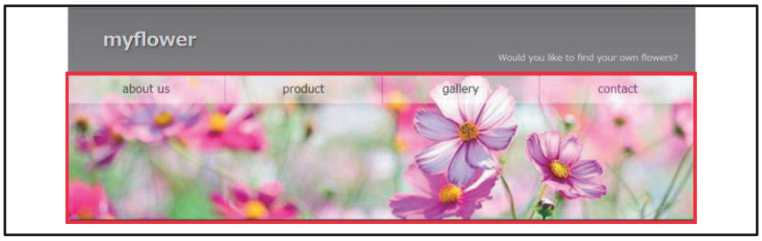
問題１（難易度 ☆）
・Webページのバナー画像を変更するプログラムを作成する
・JavaScriptプログラムは、「script」フォルダの「exercise1.js」ファイルに保存する
・プログラム作成後に「index.html」ファイルを実行し、Webブラウザに表示される実行結果を確認する

[要件]
・Webページのバナー画像を、Webページを読み込んだ時刻に合わせて変更する
・表示するバナー画像

時　刻	画像ファイルのパス
7時～10時	images/hydrangea_header.jpg
11時～14時	images/sunflower_header.jpg
15時～18時	images/osmanthus_header.jpg
19時～22時	images/cosmos_header.jpg
上 記 以 外	images/header-images.jpg

※Elementオブジェクト.style.backgroundImageプロパティに”URL(画像ファイルのパス)”を代入

[実行結果]



JavaScript

exercise1.js

```
'use strict';
const filePath = ["images/hydrangea_header.jpg", "images/sunflower_header.jpg",
                  "images/osmanthus_header.jpg", "images/cosmos_header.jpg",
                  "images/header-images.jpg"];                    //画像ファイルのパス

window.onload = function() {
  //Webページのバナー画像を、アクセス時刻に応じて変更する
  let bannerImage = document.                              ;
  let now = new Date();
  let      = Number(now.getHours());
  if(hour >= 7 && hour <= 10) {
    bannerImage[0].style.backgroundImage = "url(" + filePath[0] + ")";
  } else if(                          ) {
    bannerImage[0].style.backgroundImage = "url(" +                  + ")";
  } else if(                          ) {
    bannerImage[0].style.backgroundImage = "url(" +                  + ")";
  } else if(                          ) {
    bannerImage[0].style.backgroundImage = "url(" +                  + ")";
  } else {
    bannerImage[0].style.backgroundImage = "url(" +                  + ")";
  }
}
```

index.html

※以下のようにコードを**修正**する

```
<head>
  <script src="script/exercise1.js"></script>
  <link rel="icon" href="data:,">
  <meta charset="UTF-8">
  <title>フラワーショップ「マイフラワー」</title>
  <link rel="stylesheet" href="css/style.css" type="text/css">
</head>
  :
<header>
  <div class="header-color">
    <h1>myflower</h1>
    <p class="concept">Would you like to find your own flowers?</p>
  </div>
  <nav>
    <ul class="header">          <!-- ulタグのスタイルをJavaScriptプログラムで変更する -->
      <li><a href="javascript:void(0);" class="construction" >about us</a></li>
      <li><a href="javascript:void(0);" class="construction" >product</a></li>
      <li><a href="javascript:void(0);" class="construction" >gallery</a></li>
      <li><a href="contact.html">contact</a></li>
    </ul>
  </nav>
</header>
```

問題２（難易度 ☆☆）

・Webページの植物名とスタイルを変更するプログラムを作成する
・JavaScriptプログラムは、「script」フォルダの「exercise2.js」ファイルに保存する
・プログラム作成後に「index.html」ファイルを実行し、Webブラウザに表示される実行結果を確認する

［要件］

・Topicsの項目に表示されている植物名を「漢字」から「カタカナ」に変換する
・植物名一覧

植物名（漢字）	植物名（カタカナ）
金木犀	キンモクセイ
紫陽花	アジサイ
花水木	ハナミズキ

・TopicsとFlowerの項目に表示されている植物名のスタイルを変更する
　- 文字色　　（Elementオブジェクト.style.colorプロパティ）に ”darkgoldenrod” を代入する
　- 文字の太さ（Elementオブジェクト.style.fontWeightプロパティ）に ”bold” を代入する
　- 文字の影　（Elementオブジェクト.style.textShadowプロパティ）に ”1px 1px 2px black”を代入する

［実行結果］

exercise2.js

```
01  'use strict';
02  const kana = {金木犀 : ”キンモクセイ”, 紫陽花 : ”アジサイ”, 花水木 : ”ハナミズキ”};
03
04  window.onload = function() {
05    //Topicsの項目に表示されている植物名を「漢字」から「カタカナ」に変換する
06    //Topicsの項目に表示されている植物名のスタイルを変更する
07    let topics = document.getElementsByClassName(”topics”);
08    for(let elem of topics) {
09      let kanji = elem.innerHTML;
10      elem.innerHTML = kana[kanji];
11      styleChange(elem);
12    }
       :
```

exercise2.js（続き）

```
    :
13
14    //Flowersの項目に表示されている植物名のスタイルを変更する
15    
16
17
18
19  }
20
21  //Elementオブジェクトのスタイル（文字色・文字の太さ・文字の影）を変更するユーザ定義関数
22  function styleChange(elem) {
23    
24
25
26  }
```

index.html

※以下のようにコードを**修正**する

```
03  <head>
04    <script src="script/exercise2.js"></script>
05    <link rel="icon" href="data:,">
06    <meta charset="UTF-8">
07    <title>フラワーショップ「マイフラワー」</title>
08    <link rel="stylesheet" href="css/style.css" type="text/css">
09  </head>
    :
38  <section>
39    <h3 class="topics">金木犀</h3>  <!-- h3タグのスタイルをJavaScriptプログラムで変更する -->
40    <p>モクセイ科モクセイ属の常緑小高木樹で、モクセイ（ギンモクセイ）の変種。… </p>
41    <figure>
42      <img src="images/kinmokusei170x135.jpg" alt="金木犀の花" width="170" height="135"
            class="kinmokuseiImg">
43      <figcaption>秋に咲く金木犀の花</figcaption>
44    </figure>
45  </section>
    :
72  <figure class="flowers">      <!-- figureタグのスタイルをJavaScriptプログラムで変更する -->
73    <img src="images/hana_01.jpg" alt="アネモネの花" width="110" height="110">
74    <figcaption>アネモネ</figcaption>
75  </figure>
```

問題３（難易度 ☆☆☆）

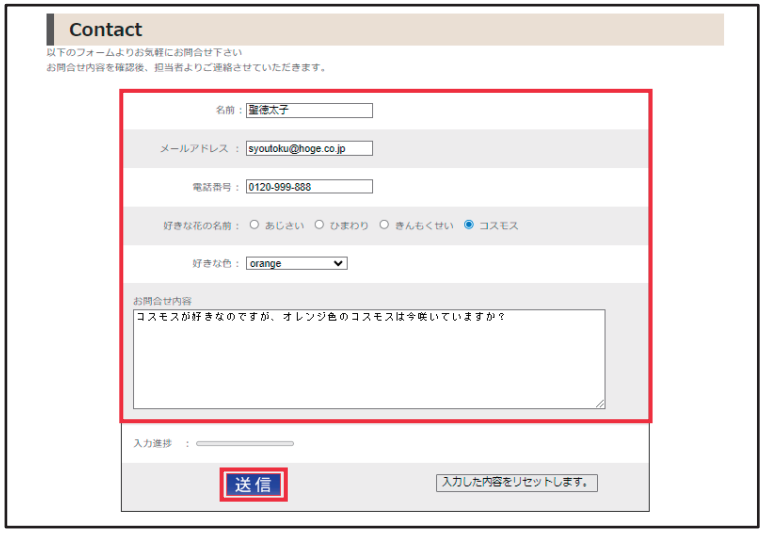
・フォームに入力した内容をコンソールに出力するプログラムを作成する
・JavaScriptプログラムは、「script」フォルダの「exercise3.js」ファイルに保存する
・プログラム作成後に「contact.html」ファイルを実行し、Webブラウザのコンソールに表示される実行結果を確認する

［要件］

・送信ボタンを押した時点で、フォームに入力した「名前」、「メールアドレス」、「電話番号」、「好きな花の名前」、「好きな色」、「お問合せ内容」を配列に格納する
・フォームの入力内容を出力する
・フォームの送信をキャンセルする

［実行結果］

（Webブラウザ）



（コンソール）

名前：聖徳太子
メールアドレス：syoutoku@hoge.co.jp
電話番号：0120-999-888
好きな花の名前：コスモス
好きな色：橙
お問合せ内容：コスモスが好きなのですが、オレンジ色のコスモスは今咲いていますか？

JavaScript

exercise3.js

```
'use strict';

function formFunction() {
  //フォームに入力した「名前」、「メールアドレス」、「電話番号」、「好きな花の名前」、「好きな色」、
  //「お問合せ内容」を配列に格納する
  let name = document.getElementsByName("name");
  let mail = document.getElementsByName("mail");
  let telephone = document.getElementsByName("telephone");
  let flower = document.getElementsByName("flower");
  let color = document.getElementsByName("color");
  let otoiawase = document.getElementsByName("otoiawase");
  const words = [];
  words.push("名前：" + name[0].value);
  words.push("メールアドレス：" + mail[0].value);
  words.push("電話番号：" + telephone [0].value);
  
  
  
  
  
  
  
  
  
  //フォームの入力内容を出力する
  for(let word of words) {
    console.log(word);
  }

  //フォームの送信をキャンセルする
  return false;
}
```

contact.html

※以下のようにコードを**修正**する

```
<head>
  <script src="script/exercise3.js"></script>
  <link rel="icon" href="data:,">
  <meta charset="UTF-8">
  <title>お問い合わせ</title>
  <link rel="stylesheet" href="css/style.css" type="text/css">
</head>
  :
```

contact.html（続き）

```
      :
31  <form action="" method="post" autocomplete="off" onsubmit="return formFunction();">
32  <p>
      <label>
        <span>名前&amp;nbsp;&amp;colon;&amp;nbsp;</span>
        <input type="text" name="name" placeholder="名前を入力してください" required>
      </label>
    </p>
33  <p>
      <label>
        <span>メールアドレス&amp;nbsp;&amp;colon;&amp;nbsp;</span>
        <input type="email" name="mail" placeholder="メールアドレスを入力してください" required>
      </label>
    </p>
34  <p>
      <label>
        <span>電話番号&amp;nbsp;&amp;colon;&amp;nbsp;</span>
        <input type="tel" name="telephone" placeholder="電話番号を入力してください" required>
      </label>
    </p>
35  <p>
      <span>好きな花の名前&amp;nbsp;&amp;colon;&amp;nbsp;</span>
      <label><input type="radio" name="flower" value="あじさい">あじさい</label>
      <label><input type="radio" name="flower" value="ひまわり">ひまわり</label>
      <label><input type="radio" name="flower" value="きんもくせい" checked>きんもくせい</label>
      <label><input type="radio" name="flower" value="コスモス">コスモス</label>
    </p>
36  <p>
      <label>
        <span>好きな色&amp;nbsp;&amp;colon;&amp;nbsp;</span>
        <select name="color" required>
37        <option value="" hidden>選択してください</option>
38        <option value="青">blue</option>
39        <option value="赤">red</option>
40        <option value="黄">yellow</option>
41        <option value="橙">orange</option>
42        <option value="灰">gray</option>
43      </select>
44    </label>
    </p>
45  <p>
      <label>お問合せ内容<br>
46      <textarea name="otoiawase" rows="10" cols="80" required></textarea>
      </label>
47  </p>
      :
53  </form>
```

問題４（難易度 ☆☆☆☆）

・Webページのバナー画像を変更するプログラムを作成する
・JavaScriptプログラムは、「script」フォルダの「exercise4.js」ファイルに保存する
・プログラム作成後に「contact.html」ファイルを実行し、Webブラウザに表示される実行結果を確認する

［要件］

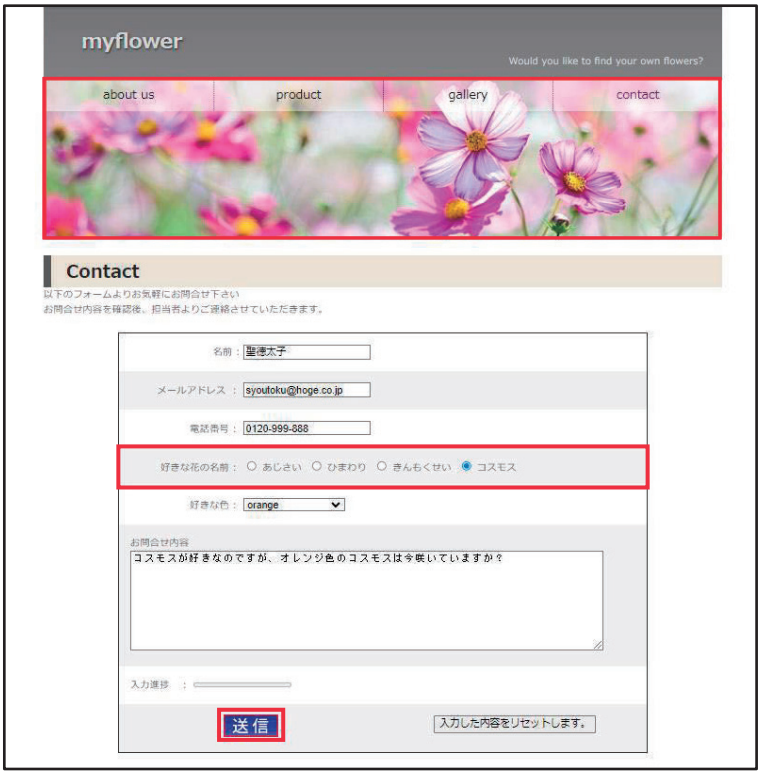
・送信ボタンを押した時点で、Webページのバナー画像を「好きな花の名前」ラジオボタンに合わせて変更する
・表示するバナー画像

花の名前	画像ファイルのパス
あじさい	images/hydrangea_header.jpg
ひまわり	images/sunflower_header.jpg
きんもくせい	images/osmanthus_header.jpg
コスモス	images/cosmos_header.jpg
その他	images/header-images.jpg

・フォームの送信をキャンセルする

［実行結果］

（コスモスを選択して送信ボタンをクリックした場合）



contact.html

※以下のようにコードを修正する

```
03  <head>
04    <script src="script/exercise4.js"></script>
05    <link rel="icon" href="data:,">
06    <meta charset="UTF-8">
07    <title>お問い合わせ</title>
08    <link rel="stylesheet" href="css/style.css" type="text/css">
09  </head>
        :
14  <header>
15    <div class="header-color">
16      <h1>myflower</h1>
17      <p class="concept">Would you like to find your own flowers?</p>
18    </div>
19    <nav>
20      <ul class="header">          <!-- ulタグのスタイルをJavaScriptプログラムで変更する -->
21        <li><a href="javascript:void(0);" class="construction" >about us</a></li>
22        <li><a href="javascript:void(0);" class="construction" >product</a></li>
23        <li><a href="javascript:void(0);" class="construction" >gallery</a></li>
24        <li><a href="contact.html">contact</a></li>
25      </ul>
26    </nav>
27  </header>
        :
31  <form action="" method="post" autocomplete="off" onsubmit="return formFunction();">
        :
35  <p>
      <span>好きな花の名前 &colon; </span>
      <label><input type="radio" name="flower" value="あじさい">あじさい</label>
      <label><input type="radio" name="flower" value="ひまわり">ひまわり</label>
      <label><input type="radio" name="flower" value="きんもくせい" checked>きんもくせい</label>
      <label><input type="radio" name="flower" value="コスモス">コスモス</label>
    </p>
        :
53  </form>
```

問題５（難易度 ☆☆☆☆☆）

・Webページの全国価格別送料一覧表を、入力された送料で更新するプログラムを作成する
・JavaScriptプログラムは、「script」フォルダの「exercise5.js」ファイルに保存する
・プログラム作成後に「admin.html」ファイルを実行し、Webブラウザに表示される実行結果を確認する

［要件］

・修正用フォームに入力された送料を取得する
・入力された送料が空欄の場合は、修正用フォームに0(円)を表示する
・全国価格別送料一覧表を更新する
　- 各セルに、入力された送料を設定する
　- 入力された送料が0(円)と一致する場合　⇒　セルに「err」クラスを追加する
　- 入力された送料が0(円)と一致しない場合　⇒　セルから「err」クラスを削除する
・フォームの送信をキャンセルする

［実行結果］

（入力例）

全国価格別送料一覧表

注文金額	本州・四国	北海道・九州	沖縄
3,000円以上	500円	700円	950円
5,000円以上	700円	850円	
10,000円以上	1,500円		
備考	購入金額が¥20,000円を超える場合は、送料が変わる場合がありますのでお問合せください。		

修正用フォーム

注文金額	本州・四国	北海道・九州	沖縄
3,000円以上	1000 円	円	3000 円
5,000円以上	円	5000 円	
10,000円以上	6000 円		

更新

（更新ボタンをクリックした場合）

全国価格別送料一覧表

注文金額	本州・四国	北海道・九州	沖縄
3,000円以上	1,000円	0円	3,000円
5,000円以上	0円	5,000円	
10,000円以上	6,000円		
備考	購入金額が¥20,000円を超える場合は、送料が変わる場合がありますのでお問合せください。		

修正用フォーム

注文金額	本州・四国	北海道・九州	沖縄
3,000円以上	1000 円	0 円	3000 円
5,000円以上	0 円	5000 円	
10,000円以上	6000 円		

更新

admin.html

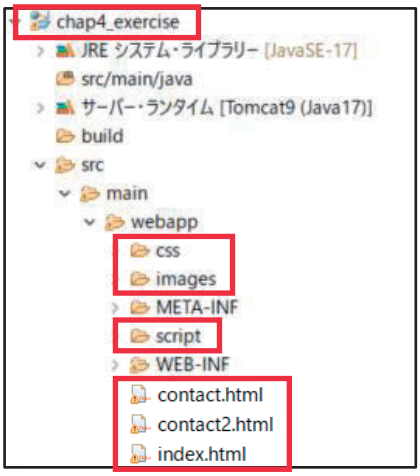
※以下のようにコードを修正する

```
03  <head>
04    <script src="script/exercise5.js"></script>
05    <link rel="icon" href="data:,">
06    <meta charset="UTF-8">
07    <title>フラワーショップ「マイフラワー」管理者用ページ</title>
08    <link rel="stylesheet" href="css/style.css" type="text/css">
09  </head>
      :
25  <table class="table-contents">   <!-- tableタグの文字列をJavaScriptプログラムで変更する -->
26    <caption>全国価格別送料一覧表</caption>
27    <thead>
28      <tr><th>注文金額</th><th>本州・四国</th><th>北海道・九州</th><th>沖縄</th></tr>
29    </thead>
30    <tbody>
31      <tr><td>3,000円以上</td><td>500円</td><td>700円</td><td rowspan="2">950円</td></tr>
32      <tr><td>5,000円以上</td><td>700円</td><td>850円</td></tr>
33      <tr><td>10,000円以上</td><td colspan="3">1,500円</td></tr>
34    </tbody>
      :
41  </table>
42  <form action="" method="post" autocomplete="off" onsubmit="return formFunction();">
43    <table class="table-update">
      :
48      <tbody>
49        <tr>
50          <td align="right">3,000円以上</td>
51          <td><input type="text" class="update">円</td>
52          <td><input type="text" class="update">円</td>
53          <td><input type="text" class="update">円</td>
54        </tr>
55        <tr>
56          <td align="right">5,000円以上</td>
57          <td><input type="text" class ="update">円</td>
58          <td cowspan="2"><input type="text" class="update">円</td>
59        </tr>
60        <tr>
61          <td align="right">10,000円以上</td>
62          <td colspan="3"><input type="text" class="update">円</td>
63        </tr>
64      </tbody>
65      <tfoot>
66        <tr>
67          <td></td>
68          <td colspan="3"><input type="submit" value="更新"></td>
69        </tr>
70      </tfoot>
71    </table>
72  </form>
```

第４章　プログラミング演習

事前準備

①Eclipse上で動的Webプロジェクトを作成する
　プロジェクト名：chap4_exercise
②配布ファイルをコピーする
　配布フォルダ名：css
　　images
　配布ファイル名：contact.html, contact2.html, index.html
③配布データを貼り付ける
　貼付先　　：webapp
④JavaScriptファイル保存用フォルダを作成する
　保存先　　：webapp
　フォルダ名　：script



次の**問題１**～**問題５**を読んで、提示された**[要件]**と**[実行結果]**を満たすプログラムを作成しなさい。

問題１（難易度 ☆）
・Webページの小さな花の画像をクリックしたときにメッセージを表示するプログラムを作成する
・JavaScriptプログラムは、「script」フォルダの「exercise1.js」ファイルに保存する
・プログラム作成後に「index.html」ファイルを実行し、Webブラウザに表示される実行結果を確認する

[要件]
・WebページのFlowerの項目に表示されている植物の画像をクリックした時点で、Webページの上部にポップアップメッセージを表示する
・表示するメッセージ

花の名前	メッセージ（花言葉）
アネモネ	赤：あなたを愛する¥n 白：真実・期待・希望¥n 紫：あなたを信じて待つ
チューリップ	赤　　：愛の告白¥n 白　　：失われた愛¥n ピンク：愛の芽生え¥n 黄　　：望みの無い恋¥n 紫　　：不滅の愛
ニッコウキスゲ	日々新たに、心安らぐ人
テッサ	人気、人望
ゼラニウム	赤　　：君が居て幸せ¥n ピンク：決意¥n 白　　：あなたの愛を信じない¥n 黄　　：予期せぬ出会い
クレマチス	精神の美、旅人の喜び、策略
ベゴニア	赤：公平¥n 白：親切
ムギワラギク	永遠の思い出、記憶

［実行結果］

（チューリップの画像をクリックした場合）

exercise1.js

```
01  'use strict';
02  const flowers
03    = { アネモネ       : "赤：あなたを愛する¥n白：真実・期待・希望¥n紫：あなたを信じて待つ",
04        チューリップ   : "赤：愛の告白¥n白：失われた愛¥nピンク：愛の芽生え"
05                       + "黄：望みの無い恋¥n紫：不滅の愛",
06        ニッコウキスゲ : "日々新たに、心安らぐ人",
07        テッサ : "人気、人望",
08        ゼラニウム     : "赤：君が居て幸せ¥nピンク：決意¥n白：あなたの愛を信じない"
09                       + "黄：予期せぬ出会い",
10        クレマチス     : "精神の美、旅人の喜び、策略",
11        ベゴニア       : "赤：公平¥n白：親切",
12        ムギワラギク   : "永遠の思い出、記憶" };                    //メッセージ（花言葉）
13
14  function flowerFunction(flower) {
15    //Webページの上部にポップアップメッセージを表示する
16    alert(          + "の花言葉¥n" +                );
17  }
```

index.html

※以下のようにコードを**修正**する

```
03  <head>
04    <script src="script/exercise1.js"></script>
05    <link rel="icon" href="data:,">
06    <meta charset="UTF-8">
07    <title>フラワーショップ「マイフラワー」</title>
08    <link rel="stylesheet" href="css/style.css" type="text/css">
09  </head>
      :
```

index.html（続き）

```
      :
 70  <div class="second-container">
 71    <figure class="flowers">
 72      <img src="images/hana_01.jpg" alt="アネモネの花"
              width="110" height="110">
 73      <figcaption>アネモネ</figcaption>
 74    </figure>
 75    <figure class="flowers">
 76      <img src="images/hana_02.jpg" alt="チューリップの花"
              width="110" height="110">
 77      <figcaption>チューリップ</figcaption>
 78    </figure>
 79    <figure class="flowers">
 80      <img src="images/hana_03.jpg" alt="ニッコウキスゲの花"
              width="110" height="110">
 81      <figcaption>ニッコウキスゲ</figcaption>
 82    </figure>
 83    <figure class="flowers">
 84      <img src="images/hana_04.jpg" alt="テッサの花"
              width="110" height="110">
 85      <figcaption>テッサ</figcaption>
 86    </figure>
 87    <figure class="flowers">
 88      <img src="images/hana_05.jpg" alt="ゼラニウムの花"
              width="110" height="110">
 89      <figcaption>ゼラニウム</figcaption>
 90    </figure>
 91    <figure class="flowers">
 92      <img src="images/hana_06.jpg" alt="クレマチス　プリンセス・ダイアナの花"
              width="110" height="110">
 93      <figcaption>クレマチス</figcaption>
 94    </figure>
 95    <figure class="flowers">
 96      <img src="images/hana_07.jpg" alt="ベゴニアの花"
              width="110" height="110">
 97      <figcaption>ベゴニア</figcaption>
 98    </figure>
 99    <figure class="flowers">
100      <img src="images/hana_08.jpg" alt="ムギワラギクの花"
              width="110" height="110">
101      <figcaption>ムギワラギク</figcaption>
102    </figure>
103  </div>
```

問題２（難易度 ☆☆）

・Webページのナビゲーションバーを制御するプログラムを作成する
・JavaScriptプログラムは、「script」フォルダの「exercise2.js」ファイルに保存する
・プログラム作成後に「index.html」ファイルを実行し、Webブラウザに表示される実行結果を確認する

［要件］

・Webページの読み込みが完了した時点で、すべてのナビゲーションボタンから「errLink」クラスを削除する
・ナビゲーションボタンをクリックすると、次の処理を実行する
  - クリックされたナビゲーションボタンに設定されているaタグの表示文字列を取得する
  - **表示文字列が ”contact” に一致しない場合**
    ⇒ **ポップアップメッセージ「申し訳ございません。このページは現在メンテナンス中です。」を表示し、aタグのclass属性に「linkErr」クラスを追加する**
  - **表示文字列が ”contact” に一致した場合**
    ⇒ **document.location.hrefプロパティに ”contact.html” を代入する**

［実行結果］

（「about us」「product」「gallery」のいずれかをクリックした場合）

（ポップアップメッセージを閉じた場合）

（「contact」をクリックした場合［ページ遷移後］）

JavaScript

exercise2.js

```
'use strict';

window.onpageshow = function() {
  //すべてのナビゲーションボタンから「errLink」クラスを削除する
  let buttons = document.getElementsByClassName("construction");
  [                                                            ]
  [                                                            ]
  [                                                            ]
}

function naviFunction(elem) {
  //クリックされたナビゲーションボタンに設定されているaタグの表示文字列を取得する
  let name = elem.innerHTML;

  //表示文字列が "contact" に一致しない場合は、ポップアップメッセージ「申し訳ございません。
  //このページは現在メンテナンス中です。」を表示し、aタグのclass属性に「linkErr」クラスを
  //追加する
  //表示文字列が "contact" に一致した場合は、document.location.hrefプロパティに
  //"contact.html" を代入する
  [                                                            ]
  [                                                            ]
  [                                                            ]
  [                                                            ]
  [                                                            ]
  [                                                            ]
}
```

index.html

※以下のようにコードを**修正**する

```
<head>
  <script src="script/exercise2.js"></script>
  <link rel="icon" href="data:,">
  <meta charset="UTF-8">
  <title>フラワーショップ「マイフラワー」</title>
  <link rel="stylesheet" href="css/style.css" type="text/css">
</head>
 :
<nav>
  <ul class="header">
    <li><a href="#" class="construction" onclick="naviFunction(this);">about us</a></li>
    <li><a href="#" class="construction" onclick="naviFunction(this);">product</a></li>
    <li><a href="#" class="construction" onclick="naviFunction(this);">gallery</a></li>
    <li><a href="#" class="construction" onclick="naviFunction(this);">contact</a></li>
  </ul>
</nav>
```

問題３（難易度 ☆☆☆）

・Webページの小さな花の画像をクリックしたときに、その画像に枠を表示するプログラムを作成する
・JavaScriptプログラムは、「script」フォルダの「exercise3.js」ファイルに保存する
・プログラム作成後に「index.html」ファイルを実行し、Webブラウザに表示される実行結果を確認する

［要件］

・Topicsの項目に表示されている植物の画像をクリックした時点で、すべての画像の境界線にnullを代入する（境界線の設定をクリアする）
・クリックされた画像の境界線に ”7px solid #00FF00” を代入する（線の太さ：7px、線種：実線、線の色：#00FF00を表示する）

［実行結果］

（金木犀の画像をクリックした場合）

（紫陽花の画像をクリックした場合）

（花水木の画像をクリックした場合）

JavaScript

exercise3.js

```
01  'use strict';
02
03  function flowerFunction2(elem) {
04    //すべての画像の境界線にnullを代入する
05    let flowers = document.getElementsByClassName("flowerImg");
06
07
08
09
10    //クリックされた画像の境界線に "7px solid #00FF00" を代入する
11
12  }
```

index.html

※以下のようにコードを**修正**する

```
03  <head>
04    <script src="script/exercise3.js"></script>
05    <link rel="icon" href="data:,">
06    <meta charset="UTF-8">
07    <title>お問い合わせ</title>
08    <link rel="stylesheet" href="css/style.css" type="text/css">
09  </head>
      :
39  <h3 class="topics">金木犀</h3>
40  <p>モクセイ科モクセイ属の常緑小高木樹で、モクセイ（ギンモクセイ）の変種。… </p>
41  <figure>
42    <img src="images/kinmokusei170x135.jpg" alt="金木犀の花" width="170" height="135"
          >
43    <figcaption>秋に咲く金木犀の花</figcaption>
44  </figure>
      :
49  <h3 class="topics">紫陽花</h3>
50  <p>アジサイ科アジサイ属の落葉低木の一種である。… </p>
51  <figure>
52    <img src="images/ajisai170x135.jpg" alt="紫陽花の花" width="170" height="135"
          >
53    <figcaption>梅雨の時期に咲く紫陽花の花</figcaption>
54  </figure>
      :
59  <h3 class="topics">花水木</h3>
60  <p>ハナミズキは、ミズキ科ミズキ属ヤマボウシ亜属の落葉高木。… </p>
61  <figure>
62    <img src="images/hanamizuki170x135.jpg" alt="花水木の花" width="170" height="135"
          >
63    <figcaption>春終わりに咲く花水木の花</figcaption>
64  </figure>
```

問題４（難易度 ☆☆☆☆）

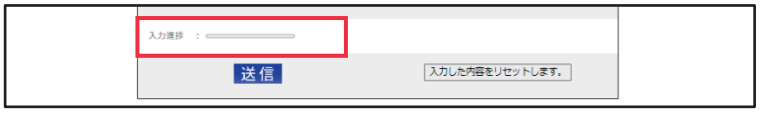
・お問合せフォームの入力状態に合わせてプログレスバーを制御するプログラムを作成する
・JavaScriptプログラムは、「script」フォルダの「exercise4.js」ファイルに保存する
・プログラム作成後に「contact.html」ファイルを実行し、Webブラウザに表示される実行結果を確認する

［要件］

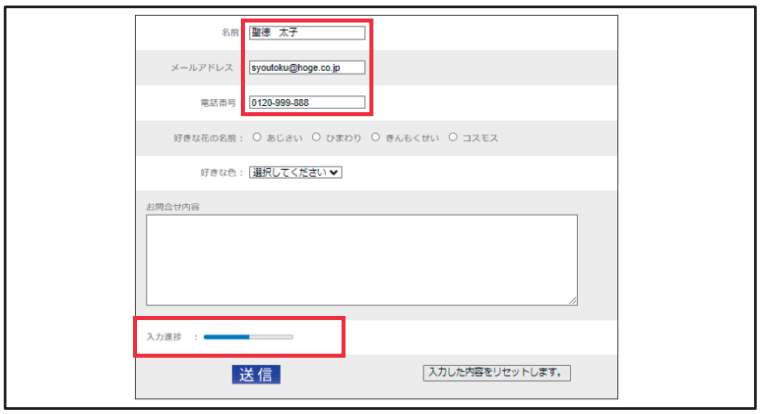
・入力フォームの内容が変更された時点で、「入力進捗」プログレスバーの値を更新する
・入力フォームの内容を変更すると、次の処理を実行する
  - **プログレスバーの値に0を代入する**
  - **氏名、メールアドレス、電話番号、お問合せ内容の文字列長が0より大きい場合は、それぞれプログレスバーの値に17%を加算する**
  - **好きな花の名前が選択されている場合は、プログレスバーの値に17%を加算する**
  - **好きな色が選択されている場合は、プログレスバーの値に15%を加算する**
  - **プログレスバーの値が100%の場合は、ポップアップメッセージ「送信準備が整いました！」を表示する**

［実行結果］

（初期状態）



（名前、メールアドレス、電話番号を入力した場合）



（名前、メールアドレス、電話番号、好きな花の名前、好きな色、お問合せ内容を入力した場合）

contact.html

※以下のようにコードを修正する

```
03  <head>
04    <meta charset="UTF-8">
05    <title>お問い合わせ</title>
06    <link rel="stylesheet" href="css/style.css" type="text/css">
07    <script src="script/exercise4.js"></script>
08    <link rel="icon" href="data:,">
09  </head>
       :
31  <form action="" method="post" autocomplete="off" onsubmit="return formFunction();">
32  <p><label>
      <span>名前 &colon; </span>
      <input type="text" name="name" placeholder="名前を入力してください"
             onchange="inputDataCheck();" required>
    </label></p>
33  <p><label>
      <span>メールアドレス &colon; </span>
      <input type="email" name="mail" placeholder="メールアドレスを入力してください"
             onchange="inputDataCheck();" required>
    </label></p>
34  <p><label>      <span>電話番号 &colon; </span>
      <input type="tel" name="telephone" placeholder="電話番号を入力してください"
             onchange="inputDataCheck();" required>
    </label></p>
35  <p><label>
      <span>好きな花の名前 &colon; </span>
      <input type="radio" name="flower" value="あじさい" onchange="inputDataCheck();" required>
      <input type="radio" name="flower" value="ひまわり" onchange="inputDataCheck();">
      <input type="radio" name="flower" value="きんもくせい" onchange="inputDataCheck();">
      <input type="radio" name="flower" value="コスモス" onchange="inputDataCheck();">
    </label></p>
36  <p><label>
      <span>好きな色 &colon; </span>
      <select name="color" onchange="inputDataCheck();" required>
37      <option hidden>選択してください</option>
38      <option value="青">blue</option>
39      <option value="赤">red</option>
40      <option value="黄">yellow</option>
41      <option value="橙">orange</option>
42      <option value="灰">gray</option>
43    </select>
44  </label></p>
45  <p><label>お問合せ内容<br>
46    <textarea name="otoiawase" … onchange="inputDataCheck();" required></textarea>
47  </label></p>
48  <p>入力進捗 &colon; <progress max="100" value="0" name="progress">0%</progress></p>
       :
53  </form>
```

問題５（難易度 ☆☆☆☆☆）

・お問合せフォームの入力書式を正規表現で確認するプログラムを作成する
・JavaScriptプログラムは、「script」フォルダの「exercise5.js」ファイルに保存する
・プログラム作成後に「contact2.html」ファイルを実行し、Webブラウザに表示される実行結果を確認する

[要件]

・「送信」ボタンを押した時点で、Webページの「氏名」、「メールアドレス」、「電話番号」の入力値の書式を正規表現で確認する
・適用する正規表現

項　目	正規表現
氏名	/^.+[　]+$/
メールアドレス	/[¥w¥d_-]+@[¥w¥d_-]+¥.[¥w¥d._-]+/
電話番号	/¥d{2,4}-¥d{3,4}-¥d{3,4}/

※「氏名」の正規表現の[ ]内に、半角空白と全角空白を指定する

・ポップアップメッセージの内容

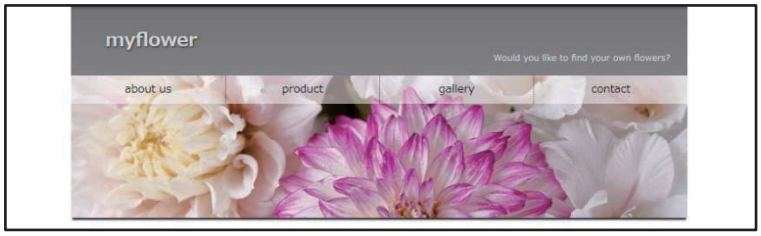
項　目	ポップアップメッセージ
氏名	氏名の書式に合わせて入力してください。¥n （例）聖徳 太子
メールアドレス	メールアドレスの書式に合わせて入力してください。¥n （例）syoutoku@hoge.co.jp”
電話番号	電話番号の書式に合わせて入力してください。¥n （例）0120-999-888

・「送信」ボタンを押すと、次の処理を実行する
- チェックに0を代入する
- 氏名の書式に一致しない場合は、ポップアップメッセージを表示し、チェックに100を加算する
- メールアドレスの書式に一致しない場合は、ポップアップメッセージを表示し、チェックに10を加算する
- 電話番号の書式に一致しない場合は、ポップアップメッセージを表示し、チェックに1を加算する
- チェックが0に等しい場合はフォームを送信し、そうでない場合はフォームの送信をキャンセルする

[実行結果]

（「氏名」の入力値が正規表現に一致しなかった場合）

（「氏名」、「メールアドレス」、「電話番号」の入力値がそれぞれ正規表現に一致した場合）



contact2.html

※以下のようにコードを修正する

```
03  <head>
04    <script src="script/exercise5.js"></script>
05    <link rel="icon" href="data:,">
06    <meta charset="UTF-8">
07    <title>フラワーショップ「マイフラワー」</title>
08    <link rel="stylesheet" href="css/style.css" type="text/css">
09  </head>
        :
31  <form action="" method="post" autocomplete="off" onsubmit="return formFunction();">
32  <p><label>
      <span>名前 &colon; </span>
      <input type="text" name="name" placeholder="名前を入力してください" required>
    </label></p>
33  <p><label>
      <span>メールアドレス &colon; </span>
      <input type="email" name="mail" placeholder="メールアドレスを入力してください" required>
    </label></p>
34  <p><label>      <span>電話番号 &colon; </span>
      <input type="tel" name="telephone" placeholder="電話番号を入力してください" required>
    </label></p>
35  <p><label>
      <span>好きな花の名前 &colon; </span>
      <input type="radio" name="flower" value="あじさい" required>
      <input type="radio" name="flower" value="ひまわり">
      <input type="radio" name="flower" value="きんもくせい">
      <input type="radio" name="flower" value="コスモス">
    </label></p>
36  <p><label>
      <span>好きな色 &colon; </span>
      <select name="color" required>
37      <option hidden>選択してください</option>
38      <option value="青">blue</option>
39      <option value="赤">red</option>
40      <option value="黄">yellow</option>
41      <option value="橙">orange</option>
42      <option value="灰">gray</option>
43    </select>
44  </label></p>
45  <p><label>お問合せ内容<br>
46    <textarea name="otoiawase" rows="10" cols="80" required></textarea>
47  </label></p>
        :
52  </form>
```

フロントエンドエンジニア養成講座

CSS/JavaScriptライブラリ

1

はじめに

【学習目標】

本テキストでは、CSS／JavaScriptライブラリを理解することを目標とします。

目次

2

CSS/JavaScriptライブラリ
第１章
Bootstrapの概要

【本章学習内容】

本章では、Bootstrapの概要について学習します。

3

第１章　Bootstrapの概要

１.１　Bootstrapとは

１.１.１　Bootstrapの特長

- Webシステム開発で広く使われている HTML/CSSのフレームワーク
- Webページのデザインに関する知識がなくても、優れたWebページを短時間で実装できる
- 「レスポンシブデザイン」に標準対応

4

第１章　Bootstrapの概要

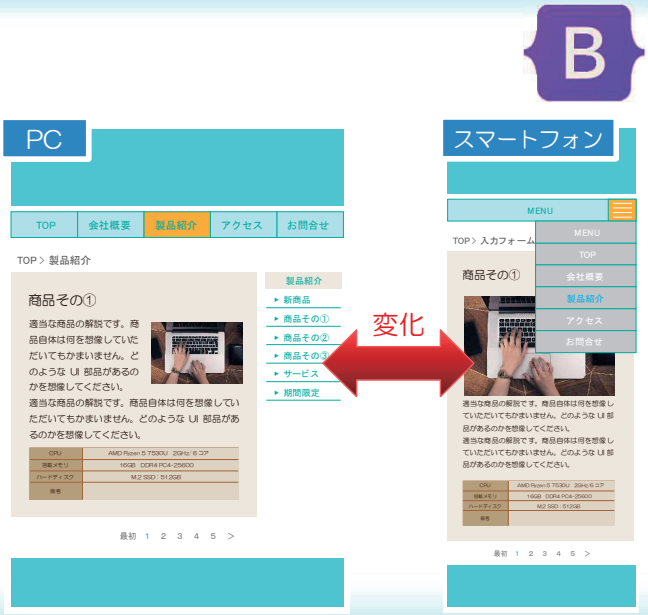
１.１　Bootstrapとは

１.１.２　レスポンシブデザイン

・ユーザのデバイスに応じてWebページの表示を変化させる仕組み

［重要視される理由］

- 検索エンジン最適化で不利にならない
- 同じWebページの表示が変化する
- Webページのメンテナス作業が短時間で行える



第１章　Bootstrapの概要

１.２　Bootstrapの利用方法

１.２.１　Bootstrapの入手

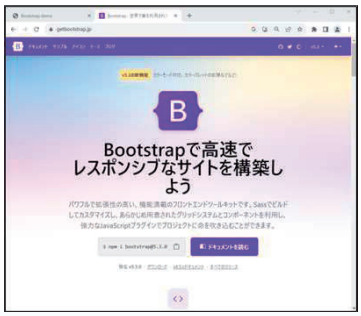
「Bootstrap公式サイト」より、Bootstrapをダウンロードします。
インターネット接続環境をご準備ください。

［ダウンロード手順］

①Webブラウザを起動

②URL欄に「https://getbootstrap.jp/」を指定

③BootstrapのWebサイトを表示



第１章　Bootstrapの概要

１.２　Bootstrapの利用方法

１.２.１　Bootstrapの入手

［ダウンロード手順］

④トップページの「ダウンロード」リンクを
クリック

7

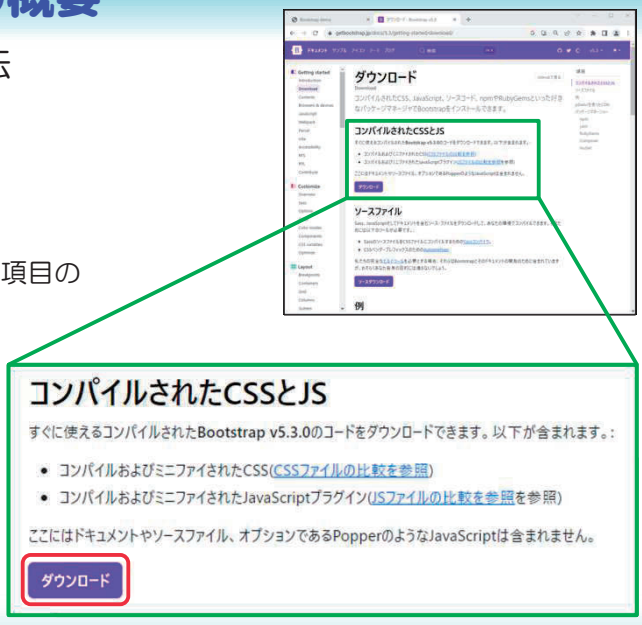
第１章　Bootstrapの概要

１.２　Bootstrapの利用方法

１.２.１　Bootstrapの入手

［ダウンロード手順］

⑤ダウンロードページの
「コンパイルされたCSSとJS」項目の
「ダウンロード」ボタンを押す



8

第１章　Bootstrapの概要

１.２　Bootstrapの利用方法

１.２.１　Bootstrapの入手

［ダウンロード手順］

⑥ダウンロードした「bootstrap-5.3.0-dist.zip」ファイルを展開（解凍）

⑦「js」フォルダ内に次のファイルがあることを確認

- bootstrap.min.js
- bootstrap.min.js.map

⑧「css」フォルダ内に次のファイルがあることを確認

- bootstrap.min.css
- bootstrap.min.css.map

第１章　Bootstrapの概要

１.２　Bootstrapの利用方法

１.２.１　Bootstrapの入手

［動的Webプロジェクトの準備］

①Eclipseを起動

②メニューの「ファイル」→「新規」→「動的Webプロジェクト」を選択

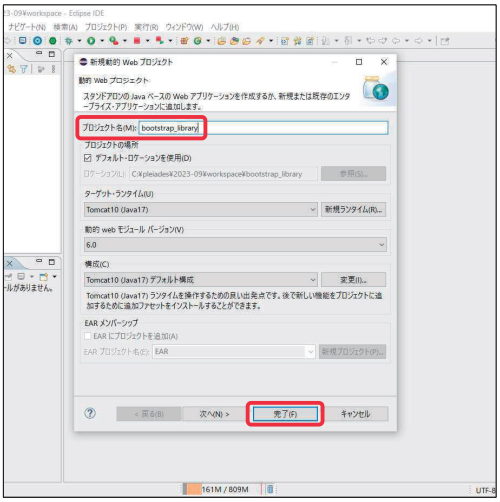
第１章　Bootstrapの概要

1.2　Bootstrapの利用方法

1.2.1　Bootstrapの入手

［動的Webプロジェクトの準備］

③プロジェクト名「bootstrap_library」を入力後、「完了」ボタンをクリック



11

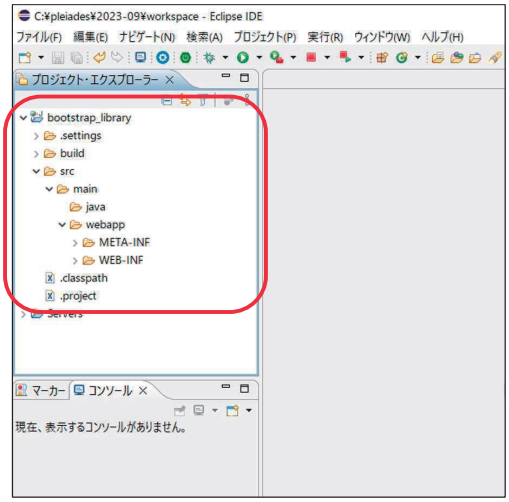
第１章　Bootstrapの概要

1.2　Bootstrapの利用方法

1.2.1　Bootstrapの入手

［動的Webプロジェクトの準備］

④「bootstrap_library」→「src」→「main」→「webapp」の順番にクリックしフォルダを展開



12

第１章　Bootstrapの概要

1.2　Bootstrapの利用方法

1.2.1　Bootstrapの入手

［動的Webプロジェクトの準備］

⑤メニューの「ファイル」→「新規」→「HTMLファイル」を選択

第１章　Bootstrapの概要

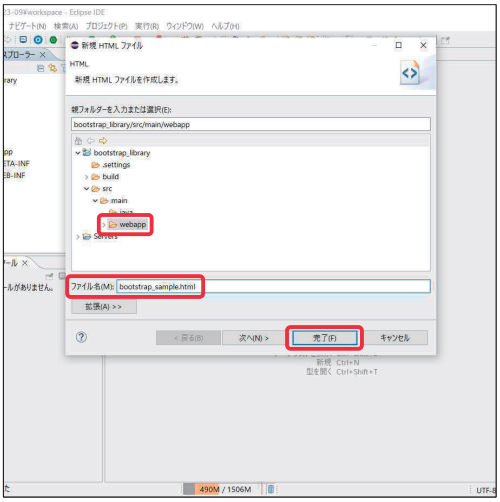
1.2　Bootstrapの利用方法

1.2.1　Bootstrapの入手

［動的Webプロジェクトの準備］

⑥保存フォルダ「webapp」を選択

⑦ファイル名「bootstrap_sample.html」を入力し「完了」ボタンをクリック



第１章　Bootstrapの概要

１.２　Bootstrapの利用方法

１.２.２　CDN経由で利用

・CDN（Contents Delivery Network）は、大容量のWebコンテンツをインターネット経由で配信するためのネットワーク

［特長］

- コンテンツ配信の高速化
- 複数サーバによる冗長構成と負荷分散
- 高度なセキュリティ対策
- 運用コストの削減　など

・CDN経由でBootstrapを利用するには

【CSS】　https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css

【JavaScript】https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js

を指定する

15

第１章　Bootstrapの概要

１.２　Bootstrapの利用方法

１.２.２　CDN経由で利用

・「bootstrap_sample.html」ファイルに下記コードを追加

```
01  <!DOCTYPE html>
02  <html lang="ja">                    <!-- HTMLファイル内に日本語が存在する -->
03  <head>
04  <meta charset="UTF-8">      <!-- ↓レスポンシブデザインの場合に必ず書く -->
05  <meta name="viewport" content="width=device-width, initial-scale=1">
06  <title>Insert title here</title>
07  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
        rel="stylesheet">
08  </head>
  :
```

16

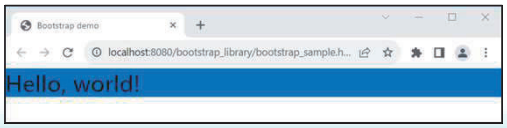
第１章　Bootstrapの概要

１.２　Bootstrapの利用方法

１.２.２　CDN経由で利用

```
   :
09  <body>                          <!-- ↓h1タグの背景色を「青」に設定する -->
10  <h1 class="bg-primary">Hello, world!</h1>
11  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js">
    </script>
12  </body>
13  </html>
```

・実行結果（Webブラウザで「bootstrap_sample.html」ファイルを表示）



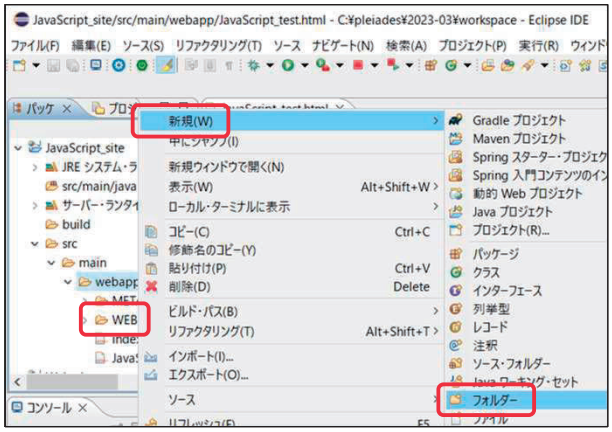
第１章　Bootstrapの概要

１.２　Bootstrapの利用方法

１.２.３　ローカルに配置して利用

[Bootstrapをローカルに配置]

①「webapp」フォルダを選択して、メニューの「ファイル」→「新規」→「フォルダー」を選択



第１章　Bootstrapの概要

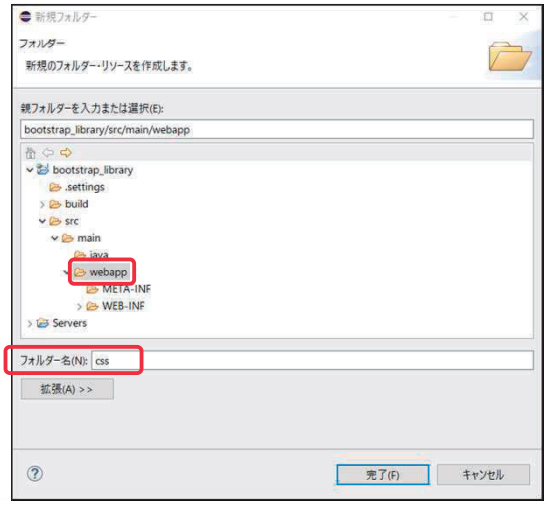
１.２　Bootstrapの利用方法

１.２.３　ローカルに配置して利用

［Bootstrapをローカルに配置］

②保存フォルダ「webapp」を選択

③フォルダ名「css」を入力し「完了」ボタンをクリック



19

第１章　Bootstrapの概要

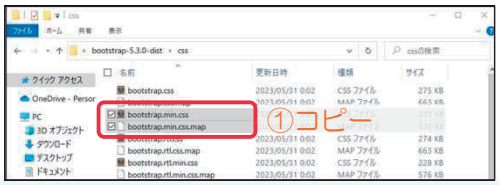
１.２　Bootstrapの利用方法

１.２.３　ローカルに配置して利用

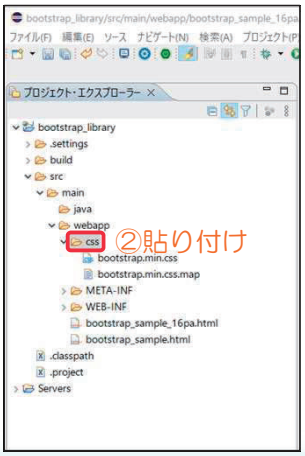
［Bootstrapをローカルに配置］

④解凍したファイル
bootstrap.min.css
bootstrap.min.css.map
をコピーし、「css」フォルダに貼り付ける

エクスプローラ



Eclipse



20

第１章　Bootstrapの概要

1.2　Bootstrapの利用方法

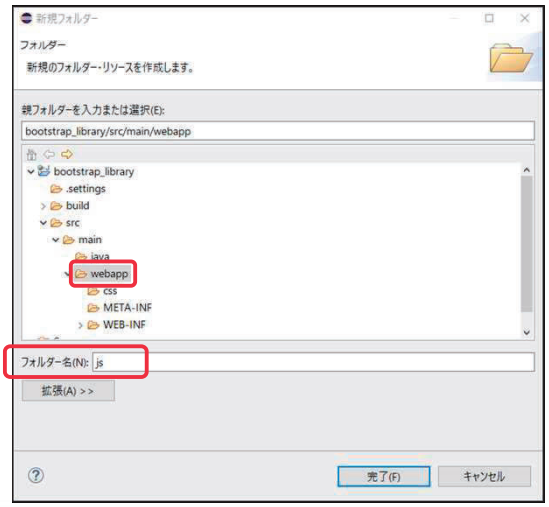
1.2.3　ローカルに配置して利用

［Bootstrapをローカルに配置］

⑤「webapp」フォルダを選択して、メニューの「ファイル」→「新規」→「フォルダー」を選択

⑥保存フォルダ「webapp」を選択

⑦フォルダ名「js」を入力し「完了」ボタンをクリック



21

第１章　Bootstrapの概要

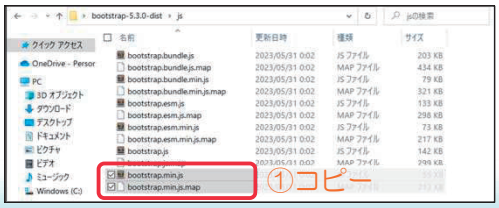
1.2　Bootstrapの利用方法

1.2.3　ローカルに配置して利用

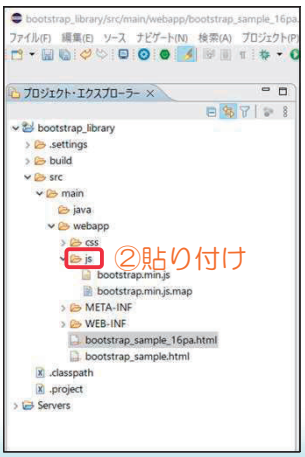
［Bootstrapをローカルに配置］

⑧解凍したファイル
　bootstrap.min.js
　bootstrap.min.js.map
をコピーし、「js」フォルダに貼り付ける

エクスプローラ



Eclipse



22

第１章　Bootstrapの概要

１.２　Bootstrapの利用方法

１.２.３　ローカルに配置して利用

・「bootstrap_sample.html」ファイルに下記コードを反映

```
01  <!DOCTYPE html>
02  <html lang="ja">                    <!-- HTMLファイル内に日本語が存在する -->
03  <head>
04  <meta charset="UTF-8">     <!-- ↓レスポンシブデザインの場合は必ず書く -->
05  <meta name="viewport" content="width=device-width, initial-scale=1">
06  <title>Insert title here</title>
07  <link href="css/bootstrap.min.css" rel="stylesheet">
08  </head>
  :
```

23

第１章　Bootstrapの概要

１.２　Bootstrapの利用方法

１.２.３　ローカルに配置して利用

```
  :
09  <body>                          <!-- ↓h1タグの背景色を「青」に設定する -->
10  <h1 class="bg-primary">Hello, world!</h1>
11  <script src="js/bootstrap.min.js"></script>
12  </body>
13  </html>
```

・実行結果（Webブラウザで「bootstrap_sample.html」ファイルを表示）

24

CSS/JavaScriptライブラリ
第２章
Bootstrap 5.3.0 の利用

【本章学習内容】

本章では、Bootstrap 5.3.0 の利用方法について学習します。

25

第２章　Bootstrap 5.3.0 の利用

2.1　Bootstrapのクラス構成

2.1.1　代表的なグループ

- レイアウト ･･･ コンテンツの配置をサポートする
  - コンテナ
  - グリッドシステム
  - 列アイテム　など
- コンテンツ ･･･ 見出し、段落、リスト、画像、テーブルなどの装飾をサポートする
  - テーブル　など
- フォーム　･･･ 入力フォーム（全体）の装飾と動作をサポートする
  - フォームコントロール
  - インプットグループ
  - フォームのレイアウト
  - 入力検証　など

26

第２章　Bootstrap 5.3.0 の利用

2.1　Bootstrapのクラス構成

2.1.1　代表的なグループ

- コンポーネント ･･･ 入力フォーム（各部品）の装飾と動作をサポートする
  - ボタン
  - カード
  - モーダル　など
- ヘルパー　　･･･ ヘッダーやフッターの配置をサポートする
  - 位置ヘルパー　など

27

第２章　Bootstrap 5.3.0 の利用

2.1　Bootstrapのクラス構成

2.1.1　代表的なグループ

- ユーティリティ ･･･ コンテンツの詳細設定をサポートする
  - 背景ユーティリティ
  - 境界ユーティリティ
  - カラーユーティリティ
  - 表示ユーティリティ
  - Flexユーティリティ
  - 不透明度ユーティリティ
  - はみ出しユーティリティ
  - サイズユーティリティ
  - 空白ユーティリティ
  - テキストユーティリティ
  - 縦の配置ユーティリティ　など

28

第２章　Bootstrap 5.3.0 の利用

2.1　Bootstrapのクラス構成

2.1.1　代表的なグループ

［学習用教材の準備］

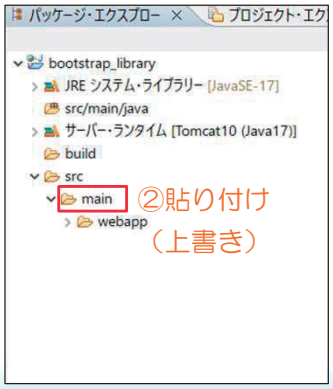
①第２章の配布プログラム「Library第２章.zip」を展開（解凍）

②「webapp」フォルダをコピー

③Eclipseで「bootstrap_library」→「src」と順番に展開し、「main」フォルダ上で貼り付け

エクスプローラ

Eclipse



29

第２章　Bootstrap 5.3.0 の利用

2.1　Bootstrapのクラス構成

2.1.1　代表的なグループ

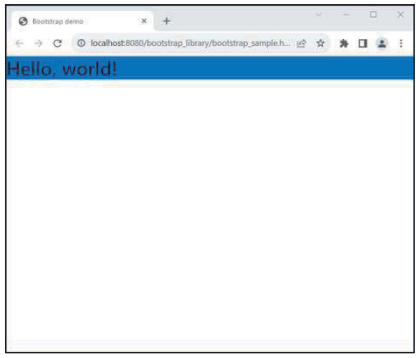
［学習用教材の準備］

④Eclipseで「main」→「webapp」と順番にを展開し、「bootstrap_sample.html」を選択

⑤メニューの「実行」→「実行」→「サーバで実行」を選択

⑥Webブラウザに「Hello, world!」が表示される

Webページ



30

第2章　Bootstrap 5.3.0 の利用

2.2　色のCSSフレームワーク

2.2.1　文字色

・Bootstrapで定義されている文字色は全24種類

［基本色と強調色］

基本色	強調色
text-primary	text-primary-emphasis
text-secondary	text-secondary-emphasis
text-success	text-success-emphasis
text-danger	text-danger-emphasis
text-warning	text-warning-emphasis
text-info	text-info-emphasis
text-light	text-light-emphasis
text-dark	text-dark-emphasis

［その他］

text-body，Black＆White
text-body
text-body-emphasis
text-body-secondary
text-body-tertiary
text-black
text-white
text-black-50
text-white-50

31

第2章　Bootstrap 5.3.0 の利用

2.2　色のCSSフレームワーク

2.2.1　文字色

・「bootstrap_sample_02_02_01.html」ファイルに下記コードを反映

```
  :
09  <body class="text-body-secondary">
10  <div><!--h1の範囲-->
11  <h1 class="text-primary-emphasis">世界の果物</h1>
12  <p>英語でfruitと言えば果実全般である … </p>
13  <p><small class="text-danger">出典: フリー百科事典 … </small></p>
14  <div><!--h2の範囲-->
15  <h2 class="text-primary">日本由来の果物</h2>
16  <p>日本に古くからある果物としては … </p>
  :
```

32

第２章　Bootstrap 5.3.0 の利用

2.2　色のCSSフレームワーク

2.2.1　文字色

```
   :
17  <div><!--h3の範囲-->
18  <h3 class="text-info-emphasis">柿</h3>
19  <p>日本では果樹として、北海道以外で広く栽培されている。… </p>
20  <p><small class="text-danger">出典: フリー百科事典 …</small></p>
21  </div><!--h3の範囲終了-->
22  </div><!--h2の範囲終了-->
23  </div><!--h1の範囲終了-->
   :
```

33

第２章　Bootstrap 5.3.0 の利用

2.2　色のCSSフレームワーク

2.2.1　文字色

・**実行結果（Webブラウザで「bootstrap_sample_02_02_01.html」ファイルを表示）**

34

第２章　Bootstrap 5.3.0 の利用

2.2　色のCSSフレームワーク

2.2.2　背景色

・Bootstrapで定義されている背景色は全22種類

［基本色と淡色］

基本色	淡　色
bg-primary	bg-primary-subtle
bg-secondary	bg-secondary-subtle
bg-success	bg-success-subtle
bg-danger	bg-danger-subtle
bg-warning	bg-warning-subtle
bg-info	bg-info-subtle
bg-light	bg-light-subtle
bg-dark	bg-dark-subtle

［その他］

text-body，Black＆White
bg-body
bg-body-secondary
bg-body-tertiary
bg-black
bg-white

透過色
bg-transparent

35

第２章　Bootstrap 5.3.0 の利用

2.2　色のCSSフレームワーク

2.2.2　背景色

・「bootstrap_sample_02_02_02.html」ファイルに下記コードを反映

```
  :
09  <body class="text-body-secondary bg-warning-subtle">
10  <div><!--h1の範囲-->
11  <h1 class="text-primary-emphasis bg-primary-subtle">世界の果物</h1>
12  <p>英語でfruitと言えば果実全般である … </p>
13  <p><small class="text-danger">出典: フリー百科事典 … </small></p>
14  <div><!--h2の範囲-->
15  <h2 class="text-primary bg-success-subtle">日本由来の果物</h2>
16  <p>日本に古くからある果物としては … </p>
  :
```

36

第２章　Bootstrap 5.3.0 の利用

2.2　色のCSSフレームワーク

2.2.2　背景色

```
  :
17  <div class="bg-light"><!--h3の範囲-->
18  <h3 class="text-info-emphasis bg-info-subtle">柿</h3>
19  <p>日本では果樹として、北海道以外で広く栽培されている。… </p>
20  <p><small class="text-danger">出典: フリー百科事典 … </small></p>
21  </div><!--h3の範囲終了-->
22  </div><!--h2の範囲終了-->
23  </div><!--h1の範囲終了-->
  :
```

37

第２章　Bootstrap 5.3.0 の利用

2.2　色のCSSフレームワーク

2.2.2　背景色

・**実行結果（Webブラウザで「bootstrap_sample_02_02_02.html」ファイルを表示 ）**

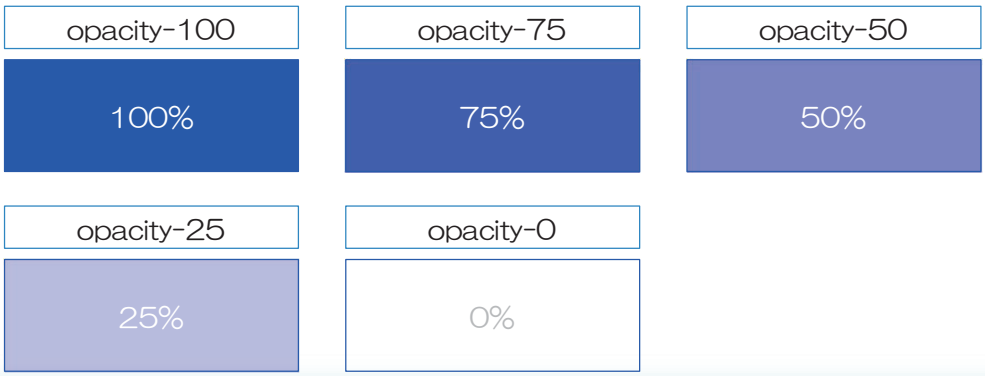
38

第2章　Bootstrap 5.3.0 の利用

2.2　色のCSSフレームワーク

2.2.3　不透明度

・Bootstrapで定義されている不透明度は5段階



39

第2章　Bootstrap 5.3.0 の利用

2.2　色のCSSフレームワーク

2.2.3　不透明度

・「bootstrap_sample_02_02_03.html」ファイルに下記コードを反映

```
  :
09  <body class="text-body-secondary bg-warning-subtle">
10  <div><!--h1の範囲-->
11  <h1 class="text-primary-emphasis bg-primary-subtle opacity-75">世界の果物</h1>
12  <p>英語でfruitと言えば果実全般である … </p>
13  <p><small class="text-danger opacity-50">出典: フリー百科事典 … </small></p>
14  <div><!--h2の範囲-->
15  <h2 class="text-primary bg-success-subtle">日本由来の果物</h2>
16  <p>日本に古くからある果物としては … </p>
  :
```

40

第２章　Bootstrap 5.3.0 の利用

2.2　色のCSSフレームワーク

2.2.3　不透明度

```
   :
17  <div class="bg-light"><!--h3の範囲-->
18  <h3 class="text-info-emphasis bg-info-subtle">柿</h3>
19  <p>日本では果樹として、北海道以外で広く栽培されている … </p>
20  <p><small class="text-danger opacity-50">出典: フリー百科事典 … </small></p>
21  </div><!--h3の範囲終了-->
22  </div><!--h2の範囲終了-->
23  </div><!--h1の範囲終了-->
   :
```

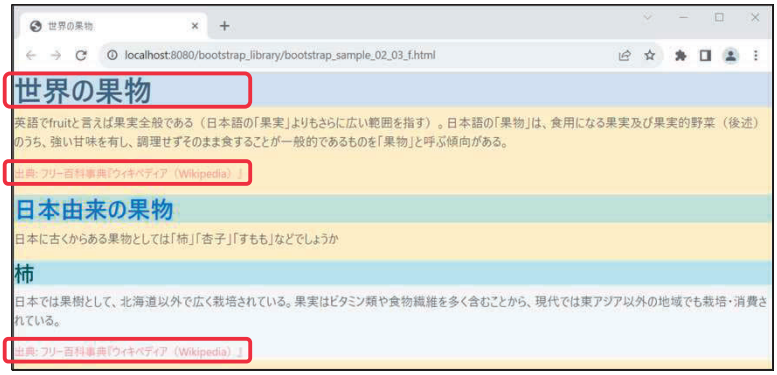
41

第２章　Bootstrap 5.3.0 の利用

2.2　色のCSSフレームワーク

2.2.3　不透明度

・**実行結果（Webブラウザで「bootstrap_sample_02_02_03.html」ファイルを表示 ）**



42

第２章　Bootstrap 5.3.0 の利用

2.3　文字のCSSフレームワーク

2.3.1　テキストの位置調整

・Bootstrapで定義されている横方向の位置調整は全３種類

左揃え

text-start　テキストの位置

中央揃え

text-center　テキストの位置

右揃え

text-end　テキストの位置

43

第２章　Bootstrap 5.3.0 の利用

2.3　文字のCSSフレームワーク

2.3.1　テキストの位置調整

・「bootstrap_sample_02_03_01.html」ファイルに下記コードを反映

```
  :
09  <body class="bg-warning-subtle text-center">
10  <div><!--h1の範囲-->
11  <h1 class="bg-primary-subtle">世界の果物</h1>
12  <p class="text-start">英語でfruitと言えば果実全般である … </p>
13  <p class="text-end"><small class="text-danger">出典: フリー百科事典 … </small></p>
14  <div><!--h2の範囲-->
15  <h2 class="bg-success-subtle">日本由来の果物</h2>
16  <p class="text-start">日本に古くからある果物としては … </p>
  :
```

44

第２章　Bootstrap 5.3.0 の利用

2.3　文字のCSSフレームワーク

2.3.1　テキストの位置調整

```
  :
17  <div class="bg-light"><!--h3の範囲-->
18  <h3 class="bg-info-subtle">柿</h3>
19  <p class="text-start">日本では果樹として、北海道以外で広く栽培されている … </p>
20  <p class="text-end"><small class="text-danger">出典: フリー百科事典 … </small></p>
21  </div><!--h3の範囲終了-->
22  </div><!--h2の範囲終了-->
23  </div><!--h1の範囲終了-->
  :
```

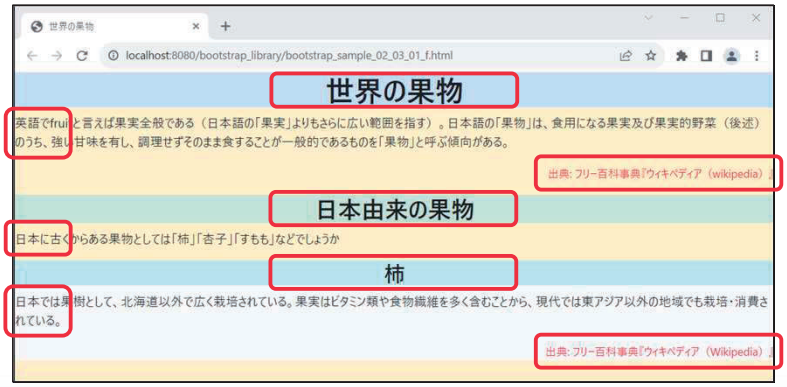
45

第２章　Bootstrap 5.3.0 の利用

2.3　文字のCSSフレームワーク

2.3.1　テキストの位置調整

・**実行結果（Webブラウザで「bootstrap_sample_02_03_01.html」ファイルを表示）**



46

第２章　Bootstrap 5.3.0 の利用

2.3　文字のCSSフレームワーク

2.3.2　フォントサイズ

・Bootstrapで定義されているフォントサイズは6段階

fs-1	テキストの大きさ
fs-2	テキストの大きさ
fs-3	テキストの大きさ
fs-4	テキストの大きさ
fs-5	テキストの大きさ
fs-6	テキストの大きさ

47

第２章　Bootstrap 5.3.0 の利用

2.3　文字のCSSフレームワーク

2.3.2　フォントサイズ

・「bootstrap_sample_02_03_02.html」ファイルに下記コードを反映

```
  :
09  <body class="bg-warning-subtle text-center fs-5">
10  <div><!--h1の範囲-->
11  <h1 class="bg-primary-subtle fs-1">世界の果物</h1>
12  <p class="text-start">英語でfruitと言えば果実全般である …</p>
13  <p class="text-end fs-6"><small class="text-danger">出典: フリー百科事典 …</small></p>
14  <div><!--h2の範囲-->
15  <h2 class="bg-success-subtle fs-3">日本由来の果物</h2>
16  <p class="text-start">日本に古くからある果物としては …</p>
  :
```

48

第２章　Bootstrap 5.3.0 の利用

2.3　文字のCSSフレームワーク

2.3.2　フォントサイズ

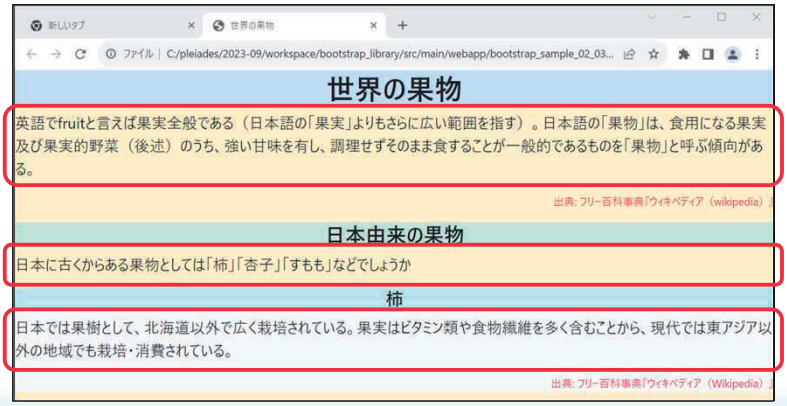
```
   :
17  <div class="bg-light"><!--h3の範囲-->
18  <h3 class="bg-info-subtle fs-4">柿</h3>
19  <p class="text-start">日本では果樹として、北海道以外で広く栽培されている …　</p>
20  <p class="text-end fs-6"><small class="text-danger">出典: フリー百科事典 …</small></p>
21  </div><!--h3の範囲終了-->
22  </div><!--h2の範囲終了-->
23  </div><!--h1の範囲終了-->
   :
```


第２章　Bootstrap 5.3.0 の利用

2.3　文字のCSSフレームワーク

2.3.2　フォントサイズ

・**実行結果（Webブラウザで「bootstrap_sample_02_03_02.html」ファイルを表示）**



第２章　Bootstrap 5.3.0 の利用

2.3　文字のCSSフレームワーク

2.3.3　フォントのスタイル

・Bootstrapで定義されているフォントのスタイルは全９種類

種　類	概　要
fw-bold	太字（太さレベル：700）
fw-bolder	太字（太さレベル：親のHTMLタグ要素の相対値）
fw-semibold	太字（太さレベル：600）
fw-medium	太字（太さレベル：500）
fw-normal	太字（太さレベル：400）［標準］
fw-light	太字（太さレベル：300）
fw-lighter	太字（太さレベル：親のHTMLタグ要素の相対値）
fst-italic	斜体（イタリック体）
fst-normal	正体［標準］

51

第２章　Bootstrap 5.3.0 の利用

2.3　文字のCSSフレームワーク

2.3.3　フォントのスタイル

・「bootstrap_sample_02_03_03.html」ファイルに下記コードを反映

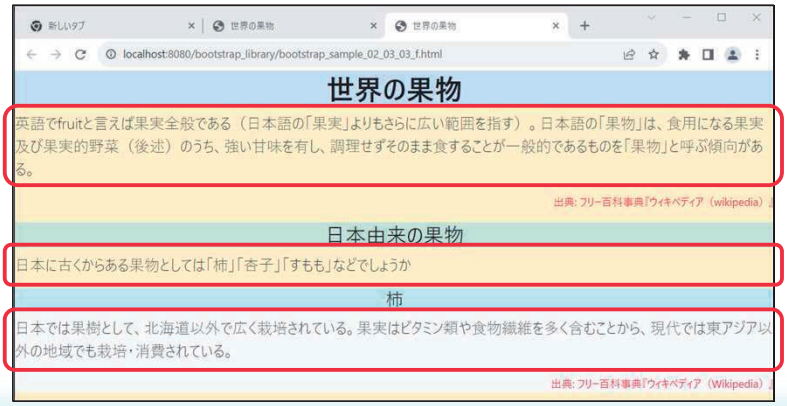
```
  :
09  <body class="bg-warning-subtle text-center fs-5 fw-light">
10  <div><!--h1の範囲-->
11  <h1 class="bg-primary-subtle fs-1">世界の果物</h1>
12  <p class="text-start">英語でfruitと言えば果実全般である … </p>
13  <p class="text-end fs-6 fw-normal">
      <small class="text-danger">出典: フリー百科事典 … </small>
    </p>
14  <div><!--h2の範囲-->
15  <h2 class="bg-success-subtle fs-3 fw-normal">日本由来の果物</h2>
16  <p class="text-start">日本に古くからある果物としては … </p>
  :
```

52

```
    :
17  <div class="bg-light"><!--h3の範囲-->
18  <h3 class="bg-info-subtle fs-4 fw-normal">柿</h3>
19  <p class="text-start">日本では果樹として、北海道以外で広く栽培されている … </p>
20  <p class="text-end fs-6 fw-normal">
      <small class="text-danger">出典: フリー百科事典 … </small>
    </p>
21  </div><!--h3の範囲終了-->
22  </div><!--h2の範囲終了-->
23  </div><!--h1の範囲終了-->
    :
```


第２章　Bootstrap 5.3.0 の利用
2.3　文字のCSSフレームワーク
2.3.3　フォントのスタイル

・実行結果（Webブラウザで「bootstrap_sample_02_03_03.html」ファイルを表示）



第２章　Bootstrap 5.3.0 の利用

2.3　文字のCSSフレームワーク

2.3.4　行の高さ

・Bootstrapで定義されている行の高さは全４種類

lh-1	テキストの行間は「１行」です。長い文章の場合は、このような感じになります。
lh-sm	テキストの行間は「1.25行」です。長い文章の場合は、このような感じになります。
lh-base	テキストの行間は「1.5行」です。長い文章の場合は、このような感じになります。
lh-lg	テキストの行間は「２行」です。長い文章の場合は、このような感じになります。

55

第２章　Bootstrap 5.3.0 の利用

2.3　文字のCSSフレームワーク

2.3.4　行の高さ

・「bootstrap_sample_02_03_04.html」ファイルに下記コードを反映

```
  :
09  <body class="bg-warning-subtle text-center fs-5 fw-light">
10  <div><!--h1の範囲-->
11  <h1 class="bg-primary-subtle fs-1">世界の果物</h1>
12  <p class= “text-start lh-1">英語でfruitと言えば果実全般である … </p>
13  <p class="text-end  fs-6 fw-normal">
      <small class="text-danger">出典: フリー百科事典 … </small>
    </p>
14  <div><!--h2の範囲-->
15  <h2 class="bg-success-subtle fs-3 fw-normal">日本由来の果物</h2>
16  <p class="text-start">日本に古くからある果物としては … </p>
  :
```

56

```
  :
17  <div class="bg-light"><!--h3の範囲-->
18  <h3 class="bg-info-subtle fs-4 fw-normal">柿</h3>
19  <p class="text-start lh-lg">日本では果樹として、北海道以外で広く栽培されている … </p>
20  <p class="text-end fs-6 fw-normal">
      <small class="text-danger">出典: フリー百科事典 … </small>
    </p>
21  </div><!--h3の範囲終了-->
22  </div><!--h2の範囲終了-->
23  </div><!--h1の範囲終了-->
  :
```


第２章　Bootstrap 5.3.0 の利用
2.3　文字のCSSフレームワーク
2.3.4　行の高さ

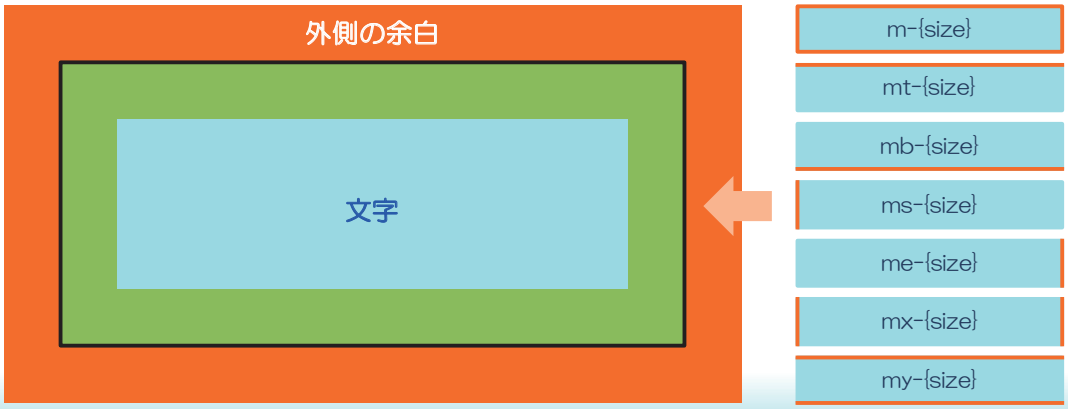
・**実行結果（Webブラウザで「bootstrap_sample_02_03_02.html」ファイルを表示）**

第2章　Bootstrap 5.3.0 の利用

2.4　ボックスレイアウト

2.4.1　マージン

・Bootstrapで定義されているマージン（外側の余白）の指定方法は全７種類



59

第2章　Bootstrap 5.3.0 の利用

2.4　ボックスレイアウト

2.4.1　マージン

・クラス名の {size} 部分はサイズ0～5、autoを指定

サイズ0	サイズ1	サイズ2	サイズ3	サイズ4	サイズ5	サイズ自動
m-0	m-1	m-2	m-3	m-4	m-5	m-auto
mt-0	mt-1	mt-2	mt-3	mt-4	mt-5	mt-auto
mb-0	mb-1	mb-2	mb-3	mb-4	mb-5	mb-auto
ms-0	ms-1	ms-2	ms-3	ms-4	ms-5	ms-auto
me-0	me-1	me-2	me-3	me-4	me-5	me-auto
mx-0	mx-1	mx-2	mx-3	mx-4	mx-5	mx-auto
my-0	my-1	my-2	my-3	my-4	my-5	my-auto

（余白なし）（最も幅狭）←→（最も幅広）

60

第２章　Bootstrap 5.3.0 の利用

2.4　ボックスレイアウト

2.4.1　マージン

・「bootstrap_sample_02_04_00.html」ファイルに下記コードを追加

```
   :
10  <body class="bg-secondary-subtle">
11  <div class="mx-auto"><!--h1の範囲-->
12  <h1 class="bg-primary-subtle text-center mx-auto mb-3">世界の果物</h1>
13  <p>英語でfruitと言えば果実全般である … </p>
14  <p><small>出典: フリー百科事典 … </small></p>
   :
```

61

第２章　Bootstrap 5.3.0 の利用

2.4　ボックスレイアウト

2.4.1　マージン

```
   :
15  <div class="bg-warning-subtle"><!--h2の範囲-->
16  <h2 class="bg-warning text-center mx-auto mb-3">日本由来の果物</h2>
17  <p>日本に古くからある果物としては … </p>
   :
```

62

第２章　Bootstrap 5.3.0 の利用

2.4　ボックスレイアウト

2.4.1　マージン

```
   :
18  <div class="bg-info-subtle"><!--h3の範囲-->
19  <h3 class="bg-info text-center mx-auto mb-3">柿</h3>
20  <p>日本では果樹として、北海道以外で広く栽培されている。… </p>
21  <p><small>出典: フリー百科事典『ウィキペディア（Wikipedia）』</small></p>
22  </div><!--h3の範囲終了-->
23  </div><!--h2の範囲終了-->
24  </div><!--h1の範囲終了-->
   :
```

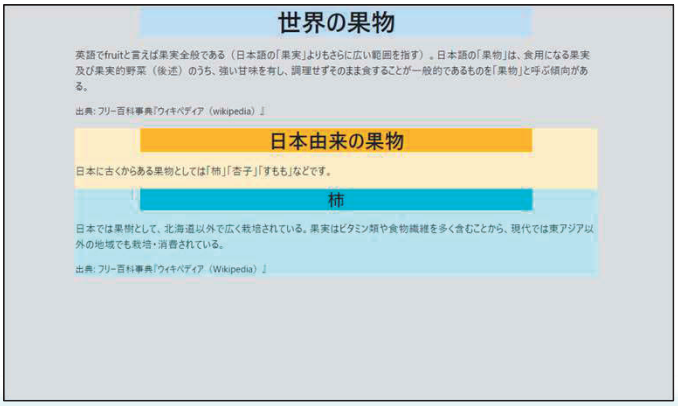
63

第２章　Bootstrap 5.3.0 の利用

2.4　ボックスレイアウト

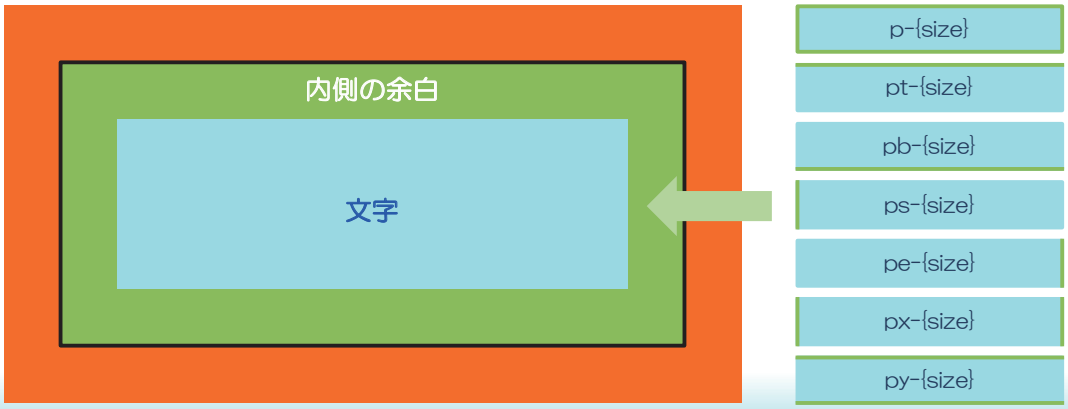
2.4.1　マージン

・**実行結果（Webブラウザで「bootstrap_sample_02_04_00.html」ファイルを表示）**



64

・Bootstrapで定義されているパディング（内側の余白）の指定方法は全７種類



・クラス名の {size} 部分はサイズ0～5、autoを指定

サイズ0	サイズ1	サイズ2	サイズ3	サイズ4	サイズ5	サイズ自動
p-0	p-1	p-2	p-3	p-4	p-5	p-auto
pt-0	pt-1	pt-2	pt-3	pt-4	pt-5	pt-auto
pb-0	pb-1	pb-2	pb-3	pb-4	pb-5	pb-auto
ps-0	ps-1	ps-2	ps-3	ps-4	ps-5	ps-auto
pe-0	pe-1	pe-2	pe-3	pe-4	pe-5	pe-auto
px-0	px-1	px-2	px-3	px-4	px-5	px-auto
py-0	py-1	py-2	py-3	py-4	py-5	py-auto

（余白なし）（最も幅狭）←→（最も幅広）

・「bootstrap_sample_02_04_00.html」ファイルに下記コードを反映

```
   :
10  <body class=" bg-secondary-subtle">
11  <div class=" mx-auto p-3"><!--h1の範囲-->
12  <h1 class="bg-primary-subtle text-center mx-auto mb-3 p-2">世界の果物</h1>
13  <p>英語でfruitと言えば果実全般である … </p>
14  <p><small>出典: フリー百科事典 … </small></p>
   :
```

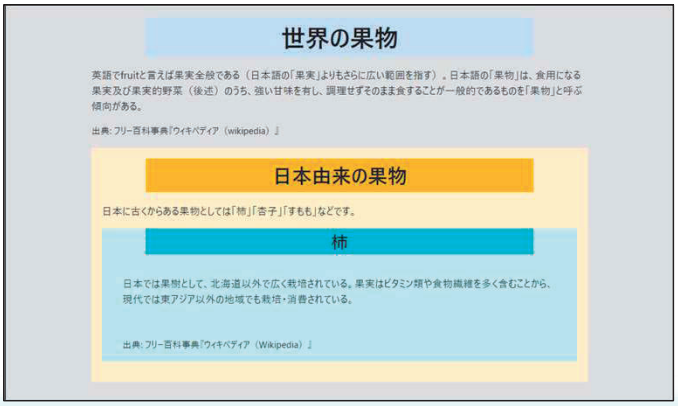


```
   :
15  <div class="bg-warning-subtle p-3"><!--h2の範囲-->
16  <h2 class="bg-warning text-center mx-auto mb-3 p-2">日本由来の果物</h2>
17  <p>日本に古くからある果物としては … </p>
   :
```

```
   :
18  <div class="bg-info-subtle px-3"><!--h3の範囲-->
19  <h3 class="bg-info text-center mx-auto mb-3 p-1">柿</h3>
20  <p class="p-3">日本では果樹として、北海道以外で広く栽培されている。… </p>
21  <p class="p-3"><small>出典: フリー百科事典 … </small></p>
22  </div><!--h3の範囲終了-->
23  </div><!--h2の範囲終了-->
24  </div><!--h1の範囲終了-->
   :
```


第２章　Bootstrap 5.3.0 の利用
2.4　ボックスレイアウト
2.4.2　パディング

・**実行結果（Webブラウザで「bootstrap_sample_02_04_00.html」ファイルを表示 ）**

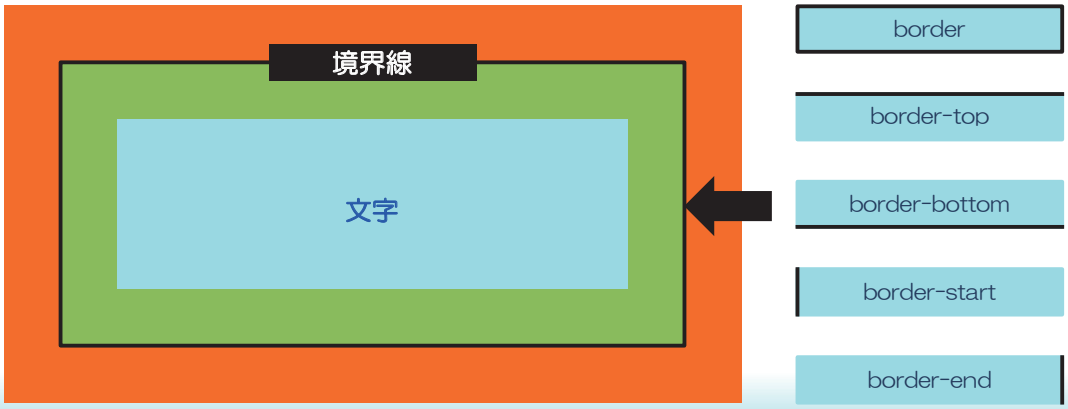


第２章　Bootstrap 5.3.0 の利用

2.4　ボックスレイアウト

2.4.3　ボーダー

・Bootstrapで定義されているボーダー（境界線）の指定方法は全５種類



71

第２章　Bootstrap 5.3.0 の利用

2.4　ボックスレイアウト

2.4.3　ボーダー

・Bootstrapで定義されているボーダー（境界線）の太さは全６種類

太さ0	太さ1	太さ2	太さ3	太さ4	太さ5
border-0	border-1	border-2	border-3	border-4	border-5
border-top-0	border-top-1	border-top-2	border-top-3	border-top-4	border-top-5
border-bottom-0	border-bottom-1	border-bottom-2	border-bottom-3	border-bottom-4	border-bottom-5
border-start-0	border-start-1	border-start-2	border-start-3	border-start-4	border-start-5
border-end-0	border-end-1	border-end-2	border-end-3	border-end-4	border-end-5

（最も細い）　（最も太い）

72

第２章　Bootstrap 5.3.0 の利用

2.4　ボックスレイアウト

2.4.3　ボーダー

・Bootstrapで定義されているボーダー（境界線）の色は全18種類

border-primary	border-primary-subtle
border-secondary	border-secondary-subtle
border-success	border-success-subtle
border-danger	border-danger-subtle
border-warning	border-warning-subtle
border-info	border-info-subtle
border-light	border-light-subtle
border-dark	border-dark-subtle
border-black	border-white

73

第２章　Bootstrap 5.3.0 の利用

2.4　ボックスレイアウト

2.4.3　ボーダー

・「bootstrap_sample_02_04_00.html」ファイルに下記コードを反映

```
  :
10  <body class=" bg-secondary-subtle">
11  <div class=" mx-auto p-3"><!--h1の範囲-->
12  <h1 class="bg-primary-subtle text-center mx-auto mb-3 p-2 border border-3
              border-secondary">
      世界の果物
    </h1>
13  <p>英語でfruitと言えば果実全般である …</p>
14  <p><small>出典: フリー百科事典 …</small></p>
  :
```

74

第２章　Bootstrap 5.3.0 の利用

2.4　ボックスレイアウト

2.4.3　ボーダー

```
  :
15  <div class="bg-warning-subtle p-3 border border-1 border-secondary"><!--h2の範囲-->
16  <h2 class="bg-warning text-center mx-auto mb-3 p-2 border border-3 border-start-0
              border-end-0 border-secondary">
      日本由来の果物
    </h2>
17  <p>日本に古くからある果物としては … </p>
  :
```


第２章　Bootstrap 5.3.0 の利用

2.4　ボックスレイアウト

2.4.3　ボーダー

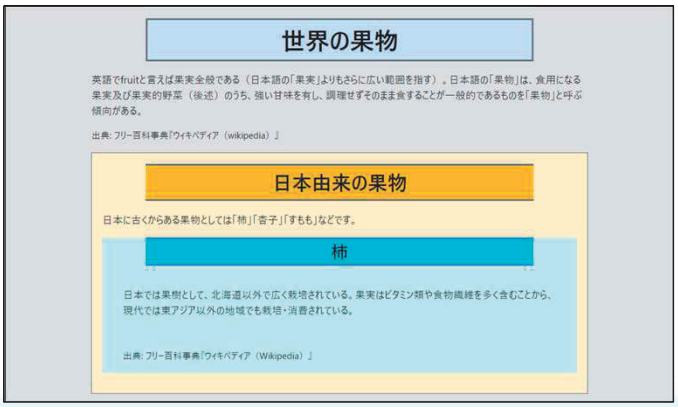

```
  :
18  <div class="bg-info-subtle px-3"><!--h3の範囲-->
19  <h3 class="bg-info text-center mx-auto mb-3 p-1 border border-2 border-start-0
              border-end-0 border-primary">
      柿
    </h3>
20  <p class="p-3">日本では果樹として、北海道以外で広く栽培されている。… </p>
21  <p class="p-3"><small>出典: フリー百科事典 … </small></p>
22  </div><!--h3の範囲終了-->
23  </div><!--h2の範囲終了-->
24  </div><!--h1の範囲終了-->
  :
```

第２章　Bootstrap 5.3.0 の利用

2.4　ボックスレイアウト

2.4.3　ボーダー

・**実行結果（Webブラウザで「bootstrap_sample_02_04_00.html」ファイルを表示）**



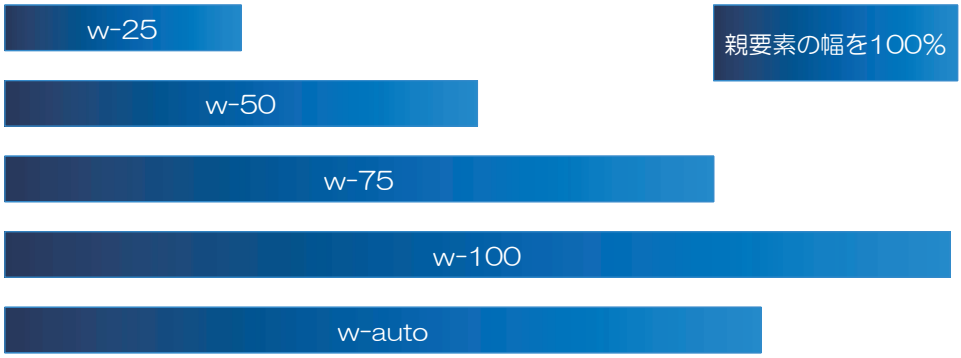
77

第２章　Bootstrap 5.3.0 の利用

2.5　レイアウトの調整

2.5.1　サイズ

・Bootstrapで定義されている、相対的な幅の指定方法は全５種類



78

第２章　Bootstrap 5.3.0 の利用

2.5　レイアウトの調整

2.5.1　サイズ

・「bootstrap_sample_02_05_00.html」ファイルに下記コードを反映

```
:
11  <div class="mx-auto p-3 w-75"><!--h1の範囲-->
:
23  </div><!--h2の範囲終了-->
24  <div class="bg-warning-subtle mt-0 p-3 border border-1 border-secondary"><!--h2の範囲-->
25  <h2 class="bg-warning text-center mx-auto mb-0 p-2 border border-3 border-start-0
              border-end-0 border-secondary w-75">日本で食べられている世界の果物</h2>
26  <p class="p-3">バナナ・キウイフルーツ・マンゴーなど、… </p>
:
```

79

第２章　Bootstrap 5.3.0 の利用

2.5　レイアウトの調整

2.5.1　サイズ

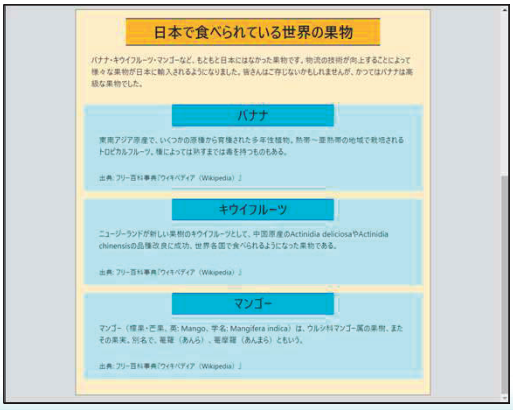
```
:
27  <div class="bg-info-subtle px-3"><!--h3の範囲-->
28  <h3 class="bg-info text-center mx-auto mb-0 p-1 border border-2 border-start-0
              border-end-0 border-primary w-50">バナナ</h3>
29  <p class="p-3 pb-0">東南アジア原産で、… </p>
30  <p class="p-3"><small>出典: フリー百科事典 … </small></p>
31  </div><!--h3の範囲終了-->
32  <div class="bg-info-subtle px-3"><!--h3の範囲-->
33  <h3 class="bg-info text-center mx-auto mb-0 p-1 border border-2 border-start-0
              border-end-0 border-primary w-50">キウイフルーツ</h3>
34  <p class="p-3 pb-0">ニュージーランドが … </p>
35  <p class="p-3"><small>出典: フリー百科事典 … </small></p>
:
```

80

```
    :
36  </div><!--h3の範囲終了-->
37  <div class="bg-info-subtle px-3"><!--h3の範囲-->
38  <h3 class="bg-info text-center mx-auto mb-0 p-1 border border-2 border-start-0
              border-end-0 border-primary w-50">マンゴー</h3>
39  <p class="p-3 pb-0">マンゴー（檬果・芒果、英: Mango、学名: Mangifera indica）は、… </p>
40  <p class="p-3"><small>出典: フリー百科事典 … </small></p>
41  </div><!--h3の範囲終了-->
42  </div><!--h2の範囲終了-->
43  </div><!--h1の範囲終了-->
    :
```


第２章　Bootstrap 5.3.0 の利用
2.5　レイアウトの調整
2.5.1　サイズ

・実行結果（Webブラウザで「bootstrap_sample_02_05_00.html」ファイルを表示）

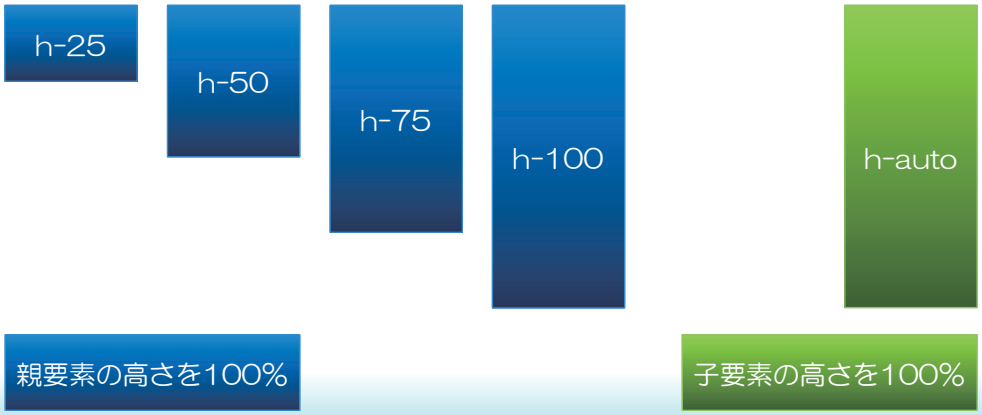


第２章　Bootstrap 5.3.0 の利用

2.5　レイアウトの調整

2.5.1　サイズ

・Bootstrapで定義されている、相対的な高さの指定方法は全５種類



83

第２章　Bootstrap 5.3.0 の利用

2.5　レイアウトの調整

2.5.1　サイズ

・「bootstrap_sample_02_05_00.html」ファイルに下記コードを反映

```
：
24  <div class="bg-warning-subtle mt-3 p-3 border border-1 border-secondary"><!--h2の範囲-->
25  <h2 class="bg-warning text-center mx-auto mb-0 p-2 border border-3 border-start-0
            border-end-0 border-secondary w-75">日本で食べられている世界の果物</h2>
26  <p class="p-3">バナナ・キウイフルーツ・マンゴーなど、… </p>
27  <div class="d-flex" style="height: 500px;">
28  <div class="bg-info-subtle px-3 me-2 h-25"><!--h3の範囲-->
29  <h3 class="bg-info text-center mx-auto mb-0 p-1 border border-2 border-start-0
            border-end-0 border-primary w-50">バナナ</h3>
30  <p class="p-3 pb-0">東南アジア原産で、… </p>
：
```

84

```
   :
31  <p class="p-3"><small>出典: フリー百科事典 … </small></p>
32  </div><!--h3の範囲終了-->
33  <div class="bg-info-subtle px-3 mx-2 h-50"><!--h3の範囲-->
34  <h3 class="bg-info text-center mx-auto mb-0 p-1 border border-2 border-start-0
              border-end-0 border-primary w-50">キウイフルーツ</h3>
35  <p class="p-3 pb-0">ニュージーランドが … </p>
36  <p class="p-3"><small>出典: フリー百科事典 … </small></p>
37  </div><!--h3の範囲終了-->
   :
```



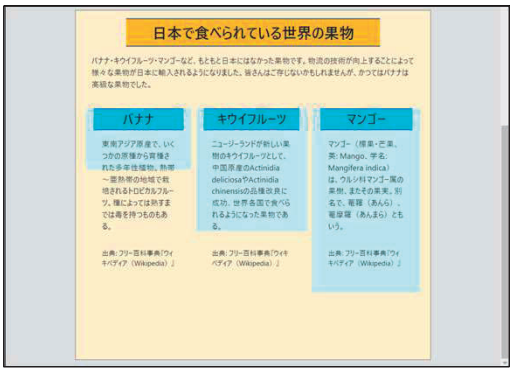
```
   :
38  <div class="bg-info-subtle px-3 ms-2 h-75"><!--h3の範囲-->
39  <h3 class="bg-info text-center mx-auto mb-0 p-1 border border-2 border-start-0
              border-end-0 border-primary w-50">マンゴー</h3>
40  <p class="p-3 pb-0">マンゴー（檬果・芒果、英: Mango、学名: Mangifera indica）は、… </p>
41  <p class="p-3"><small>出典: フリー百科事典 … </small></p>
42  </div><!--h3の範囲終了-->
43  </div>
44  </div><!--h2の範囲終了-->
   :
```

第２章　Bootstrap 5.3.0 の利用

2.5　レイアウトの調整

2.5.1　サイズ

・実行結果（Webブラウザで「bootstrap_sample_02_05_00.html」ファイルを表示 ）



87

第２章　Bootstrap 5.3.0 の利用

2.5　レイアウトの調整

2.5.1　サイズ

・「bootstrap_sample_02_05_00.html」ファイルに下記コードを反映

```
   :
24  <div class="bg-warning-subtle mt-3 p-3 border border-1 border-secondary"><!--h2の範囲-->
25  <h2 class="bg-warning text-center mx-auto mb-0 p-2 border border-3 border-start-0
            border-end-0 border-secondary w-75">日本で食べられている世界の果物</h2>
26  <p class="p-3">バナナ・キウイフルーツ・マンゴーなど、… </p>
27  <div class="d-flex h-auto" style="height: 500px;">
28  <div class="bg-info-subtle px-3 me-2 h-25"><!--h3の範囲-->
29  <h3 class="bg-info text-center mx-auto mb-0 p-1 border border-2 border-start-0
            border-end-0 border-primary">バナナ</h3>
30  <p class="p-3 pb-0">東南アジア原産で、… </p>
   :
```

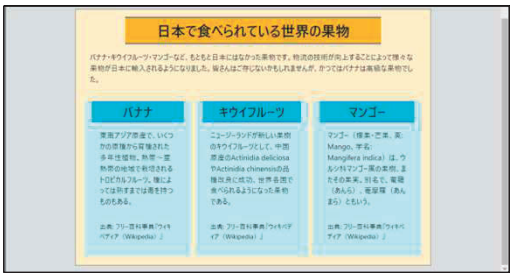
88

第２章　Bootstrap 5.3.0 の利用

2.5　レイアウトの調整

2.5.1　サイズ

・**実行結果（Webブラウザで「bootstrap_sample_02_05_00.html」ファイルを表示 ）**



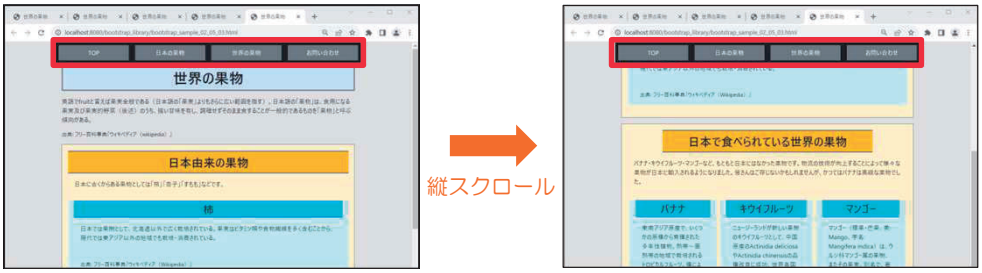
89

第２章　Bootstrap 5.3.0 の利用

2.5　レイアウトの調整

2.5.2　固定ヘッダー

・Bootstrapにおける、Webブラウザの表示領域の上部に要素を常に固定する方法



※Webページを縦スクロールしても、ヘッダーは動かない

90

・「bootstrap_sample_02_05_00.html」ファイルに下記コードを反映

```
:
11  <div class="mx-auto p-3 pt-0 w-75" style="margin-top: 60px;"><!--h1の範囲-->
12  <div class="w-75 mx-auto fixed-top">
13  <nav>
14  <ul class="bg-dark p-0">
15  <li class="w-25 text-center p-2">
      <a class="text-light bg-secondary p-2" href="#">TOP</a>
    </li>
16  <li class="w-25 text-center p-2">
      <a class="text-light bg-secondary p-2" href="#">日本の果物</a>
    </li>
:
```


第２章　Bootstrap 5.3.0 の利用
2.5　レイアウトの調整
2.5.2　固定ヘッダー

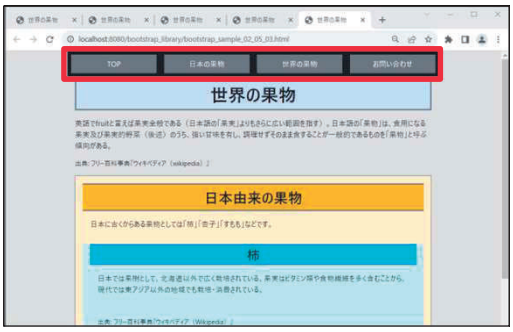
```
:
17  <li class="w-25 text-center p-2">
      <a class="text-light bg-secondary p-2" href="#">世界の果物</a>
    </li>
18  <li class="w-25 text-center p-2">
      <a class="text-light bg-secondary p-2" href="#">お問い合わせ</a>
    </li>
19  </ul>
20  </nav>
21  </div>
22  <h1 class="bg-primary-subtle text-center mx-auto mb-3 p-2 border border-3
              border-secondary">世界の果物</h1>
:
```

第２章　Bootstrap 5.3.0 の利用

2.5　レイアウトの調整

2.5.2　固定ヘッダー

・**実行結果（Webブラウザで「bootstrap_sample_02_05_00.html」ファイルを表示 ）**



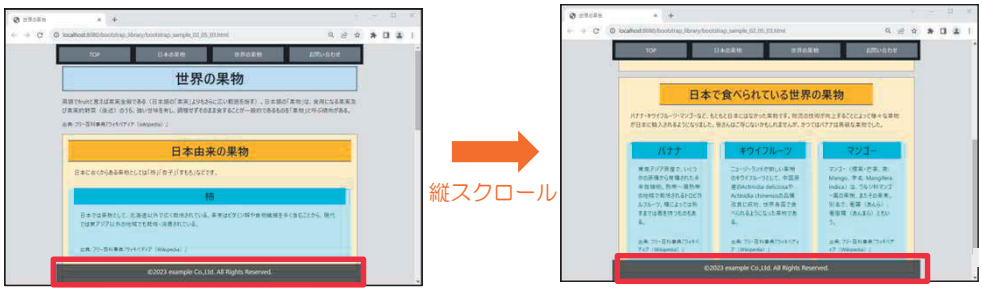
93

第２章　Bootstrap 5.3.0 の利用

2.5　レイアウトの調整

2.5.3　固定フッター

・Bootstrapにおける、Webブラウザの表示領域の下部に要素を常に固定する方法



※Webページを縦スクロールしても、フッターは動かない

94

第２章　Bootstrap 5.3.0 の利用

2.5　レイアウトの調整

2.5.3　固定フッター

・「bootstrap_sample_02_05_00.html」ファイルに下記コードを反映

```
   :
11  <div class="mx-auto p-3 pt-0 w-75"
        style="margin-top: 60px; margin-bottom: 70px;"><!--h1の範囲-->
   :
54  </div><!--h2の範囲終了-->
55  <div class="bg-dark w-75 mx-auto p-2 opacity-75 fixed-bottom">
56  <footer class="fs-5 text-center text-light">
57  <p><small>&copy;2023 example Co.,Ltd. All Rights Reserved.</small></p>
58  </footer>
59  </div>
60  </div><!--h1の範囲終了-->
   :
```


第２章　Bootstrap 5.3.0 の利用

2.5　レイアウトの調整

2.5.3　固定フッター

・**実行結果（Webブラウザで「bootstrap_sample_02_05_00.html」ファイルを表示）**

第２章　Bootstrap 5.3.0 の利用

2.5　レイアウトの調整

2.5.4　縦方向スクロールバーの常時表示

・Bootstrapにおける、スクロールバーの表示を設定する方法

※縦方向スクロールバーが表示されるWebページと表示されない
Webページが混在すると、表示の統一感が崩れてしまう

縦方向スクロールバー
表示なし

97

第２章　Bootstrap 5.3.0 の利用

2.5　レイアウトの調整

2.5.4　縦方向スクロールバーの常時表示

・「bootstrap_sample_02_05_00.html」ファイルに下記コードを反映

```
   :
10  <body class="bg-secondary-subtle overflow-y-scroll">
   :
```

・**実行結果（Webブラウザで「bootstrap_sample_02_05_00.html」ファイルを表示）**

98

第2章　Bootstrap 5.3.0 の利用

2.5　レイアウトの調整

2.5.5　テーブル

・Bootstrapにおける、テーブルの罫線とセル内の配置を設定する方法

［罫線あり］table-borderedクラス

氏名	山田　太郎
住所	大阪市北区梅田
TEL	06-555-XXXX

［罫線なし］table-borderlessクラス

氏名	山田　太郎
住所	大阪市北区梅田
TEL	06-555-XXXX

［セル内の配置］

［上揃え］ align-topクラス	山田　太郎
［上下中央揃え］ align-middleクラス	山田　太郎
［下揃え］ align-bottomクラス	山田　太郎

※左揃えはtext-startクラス、
左右中央揃えはtext-centerクラス、
右揃えはtext-endクラスを併用する

99

第2章　Bootstrap 5.3.0 の利用

2.5　レイアウトの調整

2.5.5　テーブル

・Bootstrapにおける、テーブル／行／セルの背景色を設定する方法

［背景色］

table-primary
table-secondary
table-success
table-danger
table-warning
table-info
table-light
bg-dark

［行のストライプ］table-stripedクラス

記入項目	記入例
氏名	山田　太郎
性別	男
年齢	18歳
郵便番号	555-XXXX
住所	大阪市北区梅田
電話番号	06-555-XXXX

100

第２章　Bootstrap 5.3.0 の利用

2.5　レイアウトの調整

2.5.5　テーブル

・「bootstrap_sample_02_05_00.html」ファイルに下記コードを反映

```
:
53  </div>
54  </div><!--h2の範囲終了-->
55  <div class="mt-3">
56  <table class="table table-bordered table-striped mb-0">
57  <thead class="table-success">
58  <tr class="text-center"><th>果物名</th><th>原産国</th><th>平均価格</th></tr>
59  </thead>
:
```


第２章　Bootstrap 5.3.0 の利用

2.5　レイアウトの調整

2.5.5　テーブル

```
:
60  <tbody class="table-warning">
61  <tr><td>バナナ</td><td>東南アジア</td><td class="text-end">150円</td></tr>
62  <tr><td>キウイフルーツ</td><td>中国</td><td class="text-end">150円</td></tr>
63  <tr><td>マンゴー</td><td>メキシコ</td><td class="text-end">500円</td></tr>
64  </tbody>
:
```

```
  :
65  <tfoot  class="text-end table-danger">
66  <tr><td colspan="2">平均価格</td><td>約266円</td></tr>
67  </tfoot>
68  </table>
69  </div>
70  <div class="bg-dark w-75 mx-auto p-2 opacity-75 fixed-bottom">
  :
```


・**実行結果（Webブラウザで「bootstrap_sample_02_05_00.html」ファイルを表示）**

2.6　フォーム

2.6.1　ボタン

・Bootstrapにおける、ボタンのサイズの設定方法

［基本］　　［サイズ変更］

btnクラス　　btn-lgクラス　　btn-smクラス

標準のボタン　　大きいボタン　　小さいボタン

※ボタンは境界線の色と背景色が透明に設定されている

※ボタンのサイズ変更により、フォントサイズなども変更される

105

第2章　Bootstrap 5.3.0 の利用

2.6　フォーム

2.6.1　ボタン

・Bootstrapにおける、ボタンの背景色の設定方法

btn-primary　　btn-info

btn-secondary　　btn-light

btn-success　　btn-dark

btn-danger　　btn-link

btn-warning

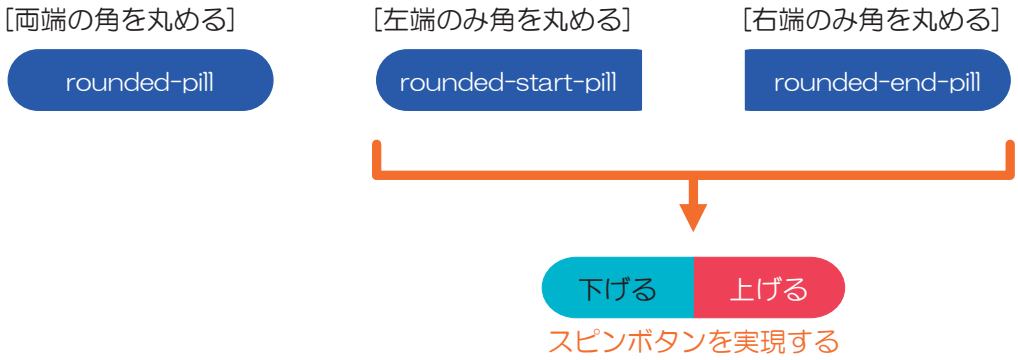
106

第2章　Bootstrap 5.3.0 の利用

2.6　フォーム

2.6.1　ボタン

・Bootstrapにおける、ボタンの角を丸める設定方法



第2章　Bootstrap 5.3.0 の利用

2.6　フォーム

2.6.1　ボタン

・「bootstrap_sample_02_06_00.html」ファイルに下記コードを反映

```
  :
10  <body>
11  <div class="p-5 mx-auto w-75 bg-warning-subtle">
12  <h1 class="fs-3 mb-4 pb-3 border-bottom">会員登録</h1>
13  <form action="" method="post">
14  <div class="mb-3">
15  <label for="name">氏名</label>
16  <input type="text" id="name" name="name" placeholder="氏名を入力">
17  </div>
  :
```

第２章　Bootstrap 5.3.0 の利用

2.6　フォーム

2.6.1　ボタン

```
  :
18  <div class="mb-3">
19  <label for="tel">電話番号</label>
20  <input type="tel" id="tel" name="tel" placeholder="電話番号をハイフンなしで入力">
21  </div>
22  <div class="mb-3">
23  <label for="address">住所</label>
24  <input type="text" id="address" name="address" placeholder="住所を入力">
25  </div>
  :
```

109

第２章　Bootstrap 5.3.0 の利用

2.6　フォーム

2.6.1　ボタン

```
  :
26  <div class="mb-5">
27  <label for="email">メールアドレス</label>
28  <input type="email" id="email" name="email" placeholder="メールアドレスを入力">
29  </div>
30  <hr>
31  <input type="submit" value="送信する" class="btn btn-primary rounded-pill">
32  <input type="reset" value="リセット" class="btn btn-secondary rounded-pill">
33  </form>
34  </div>
35  <script src="js/bootstrap.min.js"></script>
  :
```

110

第２章　Bootstrap 5.3.0 の利用

2.6　フォーム

2.6.1　ボタン

・**実行結果（Webブラウザで「bootstrap_sample_02_06_00.html」ファイルを表示）**

111

第２章　Bootstrap 5.3.0 の利用

2.6　フォーム

2.6.2　入力フォーム

・Bootstrapにおける、入力フォームのスタイルの設定方法

［設定なし］

お名前

電話番号

メールアドレス

送信

入力フォームのデザインがシンプルである

［設定あり］　form-controlクラス

お名前

電話番号

メールアドレス

送信

入力フォームの境界線の色、角を丸める・余白などが変更される

112

・「bootstrap_sample_02_06_00.html」ファイルに下記コードを反映

```
    :
15  <label for="name">氏名</label>
16  <input class="form-control" type="text" id="name" name="name"
          placeholder="氏名を入力">
    :
19  <label for="tel">電話番号</label>
20  <input class="form-control" type="tel" id="tel" name="tel"
          placeholder="電話番号をハイフンなしで入力">
    :
```



```
    :
23  <label for="address">住所</label>
24  <input class="form-control" type="text" id="address" name="address"
          placeholder="住所を入力">
    :
27  <label for="email">メールアドレス</label>
28  <input class="form-control" type="email" id="email" name="email"
          placeholder="メールアドレスを入力">
    :
```

第２章　Bootstrap 5.3.0 の利用

2.6　フォーム

2.6.2　入力フォーム

・**実行結果（Webブラウザで「bootstrap_sample_02_06_00.html」ファイルを表示）**

115

第２章　Bootstrap 5.3.0 の利用

2.6　フォーム

2.6.2　入力フォーム

・Bootstrapにおける、入力フォームのスタイルの設定方法（続き）

［設定あり］form-controlクラスを適用

お名前
電話番号
メールアドレス

送信

ラベルとフォームが
上揃えになっている

［設定あり］col-form-labelクラスを併用

お名前
電話番号
メールアドレス

送信

ラベルとフォームが
上下中央揃えになっている

116

・「bootstrap_sample_02_06_00.html」ファイルに下記コードを反映

```
   :
15  <label class="col-form-label" for="name">氏名</label>
16  <input class="form-control" type="text" id="name" name="name"
        placeholder="氏名を入力">
   :
19  <label class="col-form-label" for="tel">電話番号</label>
20  <input class="form-control" type="tel" id="tel" name="tel"
        placeholder="電話番号をハイフンなしで入力">
   :
```


第２章　Bootstrap 5.3.0 の利用
2.6　フォーム
2.6.2　入力フォーム

```
   :
23  <label class="col-form-label" for="address">住所</label>
24  <input class="form-control" type="text" id="address" name="address"
        placeholder="住所を入力">
   :
27  <label class="col-form-label" for="email">メールアドレス</label>
28  <input class="form-control" type="email" id="email" name="email"
        placeholder="メールアドレスを入力">
   :
```

第２章　Bootstrap 5.3.0 の利用

2.6　フォーム

2.6.2　入力フォーム

・**実行結果（Webブラウザで「bootstrap_sample_02_06_00.html」ファイルを表示 ）**

119

第２章　Bootstrap 5.3.0 の利用

2.6　フォーム

2.6.3　読み取り専用フォーム

・Bootstrapにおける、読み取り専用のプレーンテキストの設定方法

お名前

電話番号

メールアドレス　example@example.co.jp　平面的なデザインで、装飾を持たない

送信

120

第２章　Bootstrap 5.3.0 の利用

2.6　フォーム

2.6.3　読み取り専用フォーム

・「bootstrap_sample_02_06_00.html」ファイルに下記コードを反映

```
   :
15  <label class="col-form-label" for="name">氏名</label>
16  <input class="form-control form-control-plaintext" type="text" id="name" name="name"
        placeholder="氏名を入力" value="山田　太郎">
   :
19  <label class="col-form-label" for="tel">電話番号</label>
20  <input class="form-control form-control-plaintext" type="tel" id="tel" name="tel"
        placeholder="電話番号をハイフンなしで入力" value="1234567890">
   :
```


第２章　Bootstrap 5.3.0 の利用

2.6　フォーム

2.6.3　読み取り専用フォーム

```
   :
23  <label class="col-form-label" for="address">住所</label>
24  <input class="form-control form-control-plaintext" type="text" id="address" name="address"
        placeholder="住所を入力" value="東京都千代田区丸の内">
   :
27  <label class="col-form-label" for="email">メールアドレス</label>
28  <input class="form-control form-control-plaintext"  type="email" id="email" name="email"
        placeholder="メールアドレスを入力" value="example@example.com">
   :
31  <input type="submit" value="送信する" class="btn btn-primary rounded-pill">
32  <input type="reset" value="リセット" class="btn btn-secondary rounded-pill">
   :
```

第２章　Bootstrap 5.3.0 の利用

2.6　フォーム

2.6.3　読み取り専用フォーム

・**実行結果（Webブラウザで「bootstrap_sample_02_06_00.html」ファイルを表示）**

123

第２章　Bootstrap 5.3.0 の利用

2.6　フォーム

2.6.4　入力グループ

・Bootstrapにおける、フォームとテキストやボタンなどをグループ化する方法

メールアドレス


```
<label class="form-label" for="email">メールアドレス</label>
<div class="input-group">
  <input class="form-control" type="text" id="email" name="email1" placeholder="アカウント名">
  <span class="input-group-text">@</span>
  <input class="form-control" type="text" name="email2" placeholder="ドメイン名">
</div>
```

124

第２章　Bootstrap 5.3.0 の利用

2.6　フォーム

2.6.4　入力グループ

・「bootstrap_sample_02_06_00.html」ファイルに下記コードを反映

```
  :
14  <div class="mb-3">
15  <label class="col-form-label" for="name">氏名</label>
16  <div class="input-group">
17  <span class="input-group-text">フルネーム</span>
18  <input class="form-control-plaintext form-control" type="text" id="name" name="name"
        value="山田　太郎" placeholder="氏名を入力">
19  </div>
20  </div>
  :
```

125

第２章　Bootstrap 5.3.0 の利用

2.6　フォーム

2.6.4　入力グループ

```
  :
21  <div class="mb-3">
22  <label class="col-form-label" for="tel">電話番号</label>
23  <div class="input-group">
24  <input class="form-control-plaintext form-control" type="text" id="tel" name="tel1"
        value="1234567890" placeholder="00">
25  <span class="input-group-text">-</span>
26  <input class="form-control" type="text" name="tel2" placeholder="0000">
27  <span class="input-group-text">-</span>
28  <input class="form-control" type="text" name="tel3" placeholder="0000">
29  </div>
30  </div>
  :
```

126

```
  :
31  <div class="mb-3">
32  <label class="col-form-label" for="address">住所</label>
33  <div class="input-group">
34  <span class="input-group-text">都道府県から</span>
35  <input class="form-control-plaintext form-control" type="text" id="address" name="address"
        value="example@example.com" placeholder="住所を入力">
36  </div>
37  </div>
  :
```


第2章　Bootstrap 5.3.0 の利用
2.6　フォーム
2.6.4　入力グループ

```
  :
38  <div class="mb-3">
39  <label class="col-form-label" for="email">メールアドレス</label>
40  <div class="input-group">
41  <input class="form-control-plaintext form-control" type="text" id="email" name="email1"
        value="example@example.com" placeholder="アカウント名">
42  <span class="input-group-text">@</span>
43  <input class="form-control" type="text" name="email2" placeholder="ドメイン名">
44  </div>
45  </div>
  :
```

第２章　Bootstrap 5.3.0 の利用

2.6　フォーム

2.6.4　入力グループ

・**実行結果（Webブラウザで「bootstrap_sample_02_06_00.html」ファイルを表示）**

129

第２章　Bootstrap 5.3.0 の利用

2.6　フォーム

2.6.5　カスタムフィードバックスタイル

・Bootstrapにおける、入力チェックとフィードバックの仕組みをカスタマイズする方法

［フィードバックスタイル設定なし］

お名前　山田　太郎

電話番号　（未入力）

メールアドレス　（未入力）

送信　送信ボタンを押すと、入力データが送信される

※未入力のフォームは空文字列を送信する

［カスタムフィードバックスタイル設定あり］

お名前　山田　太郎 ✓

電話番号　（未入力） ⓘ
電話番号は必須項目です。

メールアドレス　（未入力） ⓘ
メールアドレスは必須項目です。

送信　送信ボタンを押すと、未入力がある場合は送信がキャンセルされる

130

第2章　Bootstrap 5.3.0 の利用

2.6　フォーム

2.6.5　カスタムフィードバックスタイル

・「bootstrap_sample_02_06_00.html」ファイルに下記コードを反映

```
   :
13  <form class="needs-validation" action="" method="post" novalidate>
14  <div class="mb-3">
15  <label class="col-form-label" for="name">氏名</label>
16  <div class="input-group has-validation">
17  <span class="input-group-text">フルネーム</span>
18  <input class="form-control" … placeholder="氏名を入力" pattern="^.+$" required>
19  <div class="invalid-feedback">氏名は必須項目です。</div>
20  </div>
21  </div>
   :
```

131

第2章　Bootstrap 5.3.0 の利用

2.6　フォーム

2.6.5　カスタムフィードバックスタイル

```
   :
22  <div class="mb-3">
23  <label class="col-form-label" for="tel">電話番号</label>
24  <div class="input-group has-validation">
25  <input class="form-control" … placeholder="00" pattern="^¥d{2,4}$" required>
26  <span class="input-group-text">-</span>
27  <input class="form-control" … placeholder="0000" pattern="¥d{2,4}$" required>
28  <span class="input-group-text">-</span>
29  <input class="form-control" … placeholder="0000" pattern="¥d{3,4}$" required>
30  <div class="invalid-feedback">電話番号は必須項目です。</div>
31  </div>
32  </div>
   :
```

132

```
   :
33  <div class="mb-3">
34  <label class="col-form-label" for="address">住所</label>
35  <div class="input-group has-validation">
36  <span class="input-group-text">都道府県から</span>
37  <input class="form-control" … placeholder="住所を入力" pattern="^.+$" required>
38  <div class="invalid-feedback">住所は必須項目です。</div>
39  </div>
40  </div>
41  <div class="mb-5">
42  <label class="col-form-label" for="email">メールアドレス</label>
   :
```



```
   :
43  <div class="input-group has-validation">
44  <input class="form-control" … placeholder="アカウント名"
        pattern="^[a-z][a-z0-9¥-_.]*$" required>
45  <span class="input-group-text">@</span>
46  <input class="form-control" … placeholder="ドメイン名"
        pattern="^[a-z][a-z0-9¥-.]*¥.[a-z]{2,3}$" required>
47  <div class="invalid-feedback">メールアドレスは必須項目です。</div>
48  </div>
49  </div>
   :
```

第２章　Bootstrap 5.3.0 の利用

2.6　フォーム

2.6.5　カスタムフィードバックスタイル

```
  :
50  <hr>
51  <input type="submit" value="送信する" class="btn btn-primary rounded-pill">
52  </form>
53  </div>
54  <script src="js/bootstrap.min.js"></script>
55  <script src="js/validation.js"></script>
56  </body>
57  </html>
```

135

第２章　Bootstrap 5.3.0 の利用

2.6　フォーム

2.6.5　カスタムフィードバックスタイル

・**実行結果（Webブラウザで「bootstrap_sample_02_06_00.html」ファイルを表示 ）**

136

第2章　Bootstrap 5.3.0 の利用

2.7　デザインレイアウト

2.7.1　コンテナ

・コンテナは、内部のコンテンツに余白や配置を設定するためのレイアウト要素（大枠）

- containerクラス　　：画面の横幅に応じて段階的に広がるレイアウト

［画面の横幅に対するコンテナの横幅］

クラス	画面の横幅 ～575px	画面の横幅 ～767px	画面の横幅 ～991px	画面の横幅 ～1,199px	画面の横幅 ～1,399px	画面の横幅 1,400px～
container container-sm	100%	540px	720px	960px	1,140px	1,320px
container-md	100%	100%	720px	960px	1,140px	1,320px
container-lg	100%	100%	100%	960px	1,140px	1,320px
container-xl	100%	100%	100%	100%	1,140px	1,320px
container-xxl	100%	100%	100%	100%	100%	1,320px

- container-fluidクラス：画面の横幅いっぱいに広がるレイアウト（常時100%）

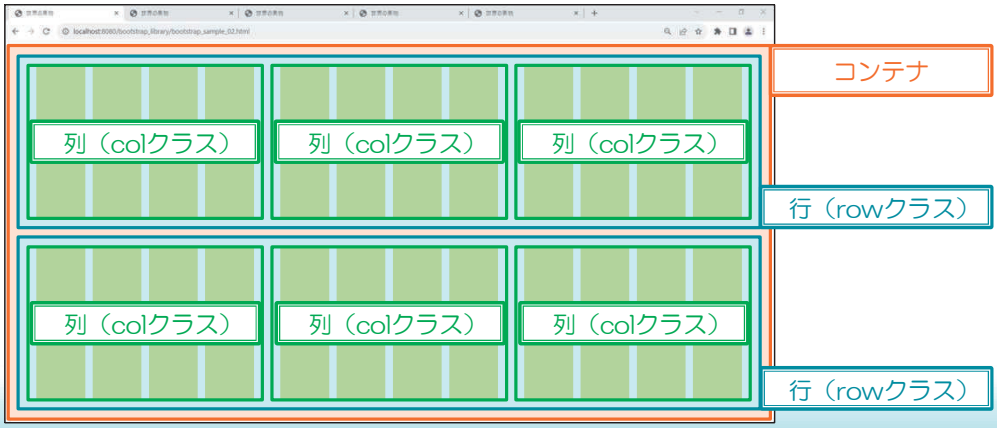
137

第2章　Bootstrap 5.3.0 の利用

2.7　デザインレイアウト

2.7.1　コンテナ

・Bootstrapでは、行と列でコンテンツを配置するグリッドシステムが利用できる



138

・「bootstrap_sample_02_07_00.html」ファイルに下記コードを反映

```
   :
11  <div style="width: 990px;" class="mx-auto"><!--大枠の範囲-->
12  <header>
13  <div class="bg-black bg-opacity-75 mx-auto"><!--navの範囲-->
14  <nav>
15  <div class="container">
16  <ul class="row text-center p-2">
17  <li class="bg-warning mx-1 py-4 col"><a href="#">TOP</a></li>
18  <li class="bg-warning mx-1 py-4 col"><a href="#">ネコ科の肉食獣とは</a></li>
   :
```



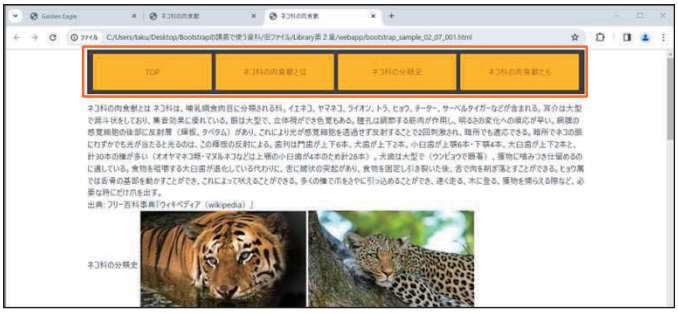
```
   :
19  <li class="bg-warning mx-1 py-4 col"><a href="#">ネコ科の分類史</a></li>
20  <li class="bg-warning mx-1 py-4 col"><a href="#">ネコ科の肉食獣たち</a></li>
21  </ul>
22  </div>
23  </nav>
24  </div><!--navの範囲-->
25  </header>
```

第２章　Bootstrap 5.3.0 の利用

2.7　デザインレイアウト

2.7.1　コンテナ

・**実行結果（Webブラウザで「bootstrap_sample_02_07_00.html」ファイルを表示）**



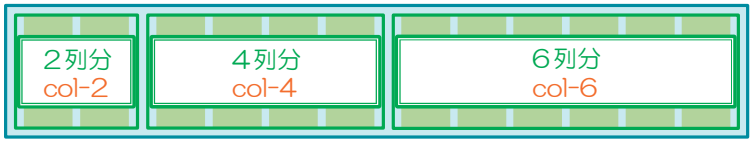
第２章　Bootstrap 5.3.0 の利用

2.7　デザインレイアウト

2.7.1　コンテナ

・コンテンツの設置は、最大12列分で計算する

・col-1クラス（１列分）～ col-12クラス（12列分）が指定できる



第２章　Bootstrap 5.3.0 の利用

2.7　デザインレイアウト

2.7.1　コンテナ

・「bootstrap_sample_02_07_00.html」ファイルに下記コードを反映

```
  :
25  </header>
26  <div class="container fs-5 text-danger my-5"><!--h1の構造の範囲-->
27  <h1 class="fs-2 text-center border border-start-0 border-end-0 py-3 my-3">
      ネコ科の肉食獣とは
    </h1>
28  <div class="row">
29  <div class="col-4"><img src="imgs/lion-300.jpg"></div>
30  <div class="col-8 text-danger-emphasis">
  :
```


第２章　Bootstrap 5.3.0 の利用

2.7　デザインレイアウト

2.7.1　コンテナ

```
  :
31  <p class="mb-3">ネコ科は、哺乳綱食肉目に分類される科。… </p>
32  <p><small>出典: フリー百科事典『ウィキペディア（Wikipedia）』</small></p>
33  </div>
34  </div>
35  </div><!--h1の構造の範囲-->
  :
```

第２章　Bootstrap 5.3.0 の利用

2.7　デザインレイアウト

2.7.1　コンテナ

・**実行結果（Webブラウザで「bootstrap_sample_02_07_00.html」ファイルを表示 ）**

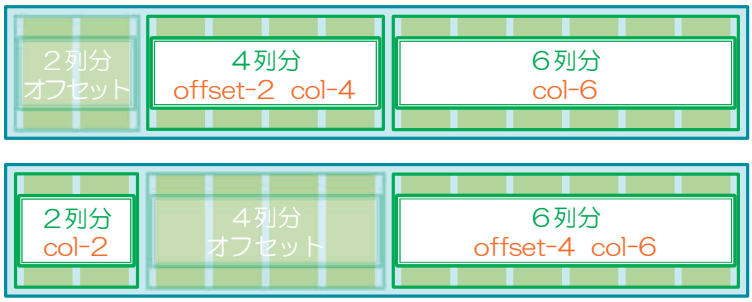
第２章　Bootstrap 5.3.0 の利用

2.7　デザインレイアウト

2.7.1　コンテナ

・コンテンツの左側にオフセット（空白の列）を挿入できる

・offset-1クラス（１列分の空白）～ offset-11クラス（11列分の空白）が指定できる



第２章　Bootstrap 5.3.0 の利用

2.7　デザインレイアウト

2.7.1　コンテナ

・「bootstrap_sample_02_07_00.html」ファイルに下記コードを反映

```
 :
30  <div class="col-8 text-danger-emphasis">
31  <p class="mb-3">ネコ科は、哺乳綱食肉目に分類される科。… </p>
32  <div class="row">
33  <p class="offset-4 col-8 p-2 bg-danger-subtle text-center">
      <small>出典: フリー百科　事典『ウィキペディア（Wikipedia）』</small>
    </p>
34  </div>
35  </div>
 :
```


第２章　Bootstrap 5.3.0 の利用

2.7　デザインレイアウト

2.7.1　コンテナ

・**実行結果（Webブラウザで「bootstrap_sample_02_07_00.html」ファイルを表示）**

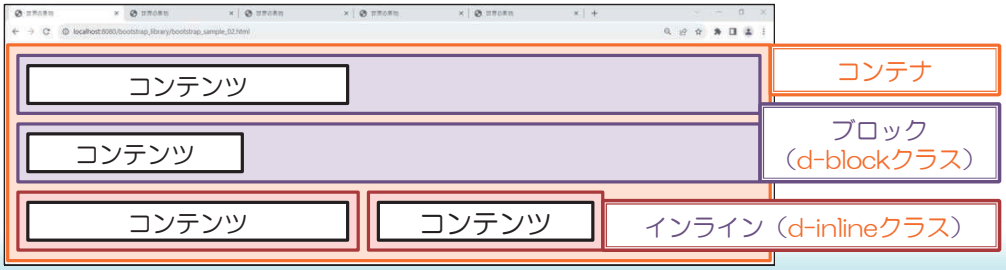
第2章　Bootstrap 5.3.0 の利用

2.7　デザインレイアウト

2.7.2　ディスプレイ

・Bootstrapでは、コンテンツの位置や並びを細かく制御できる

- ブロック　：コンテナの横幅いっぱいに広がるレイアウト要素（中枠）
  レイアウト要素の後ろに改行が入る
- インライン：コンテンツの横幅に応じて広がるレイアウト要素（中枠）
  レイアウト要素の後ろに改行が入らない



149

第2章　Bootstrap 5.3.0 の利用

2.7　デザインレイアウト

2.7.2　ディスプレイ

・「bootstrap_sample_02_07_00.html」ファイルに下記コードを反映

```
   :
37  </div><!--h1の構造の範囲-->
38  <div class="fs-5 text-danger my-5"><!--h2の範囲-->
39  <h2 class="fs-3 text-center border border-start-0 border-end-0 py-3 my-3">
      ネコ科の分類史
    </h2>
40  <div class="d-block"><img src="imgs/tiger-330.jpg"></div>
41  <div class="d-block"><img src="imgs/leopard-330.jpg"></div>
42  <div class="d-inline"><img src="imgs/puma-330.jpg"></div>
43  <p class="d-inline mb-3 text-danger-emphasis">リンネの『自然の体系』で … </p>
   :
```

150

```
   :
44   <p class="d-inline">
       <small>出典: フリー百科事典『ウィキペディア（Wikipedia）』</small>
     </p>
45   </div><!--h2の範囲終了-->
   :
```


・**実行結果（Webブラウザで「bootstrap_sample_02_07_00.html」ファイルを表示）**

第2章　Bootstrap 5.3.0 の利用

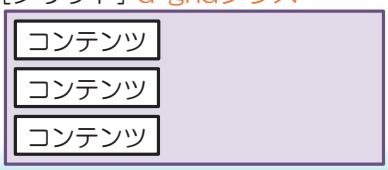
2.7　デザインレイアウト

2.7.2　ディスプレイ

・Bootstrapでは、コンテンツの位置や並びを細かく制御できる（続き）

- グリッド　：コンテナの横幅いっぱいに広がるレイアウト要素（中枠）
  内部のコンテンツを縦に（上から下に向かって）並べる
- フレックス　：コンテナの横幅いっぱいに広がるレイアウト要素（中枠）
  内部のコンテンツを横に（左から右に向かって）並べる
- インラインフレックス：コンテンツの横幅に応じて広がるレイアウト要素（中枠）
  内部のコンテンツを横に（左から右に向かって）並べる

[グリッド] d-gridクラス



[フレックス] d-flexクラス

[インラインフレックス] d-inline-flexクラス

153

第2章　Bootstrap 5.3.0 の利用

2.7　デザインレイアウト

2.7.2　ディスプレイ

・「bootstrap_sample_02_07_00.html」ファイルに下記コードを反映

```
  :
38  <div class="fs-5 text-danger my-5"><!--h2の範囲-->
39  <h2 class="fs-3 text-center border border-start-0 border-end-0 py-3 my-3">
      ネコ科の分類史
    </h2>
40  <div class="d-grid mb-3">
41  <div class="d-block"><img src="imgs/tiger-330.jpg"></div>
42  <div class="d-block"><img src="imgs/leopard-330.jpg"></div>
43  <div class="d-inline"><img src="imgs/puma-330.jpg"></div>
44  </div>
  :
```

154

```
   :
45  <p class="d-inline mb-3 text-danger-emphasis">リンネの『自然の体系』で … </p>
46  <p class="d-inline text-danger-emphasis">
      <small>出典: フリー百科事典『ウィキペディア（Wikipedia）』</small>
    </p>
47  </div><!--h2の範囲終了-->
   :
```


第２章　Bootstrap 5.3.0 の利用
2.7　デザインレイアウト
2.7.2　ディスプレイ

・**実行結果（Webブラウザで「bootstrap_sample_02_07_00.html」ファイルを表示）**

第２章　Bootstrap 5.3.0 の利用

2.7　デザインレイアウト

2.7.2　ディスプレイ

・Bootstrapでは、コンテンツの位置や並びを細かく制御できる（続き）

- フレックスの方向
  - ＊ flex-rowクラス　　：内部のコンテンツを横に（左から右に向かって）並べる
  - ＊ flex-columnクラス：内部のコンテンツを縦に（上から下に向かって）並べる
- フレックスの位置揃え
  - ＊ justify-content-startクラス　：内部のコンテンツを左揃えにする
  - ＊ justify-content-centerクラス：内部のコンテンツを左右中央揃えにする
  - ＊ justify-content-endクラス　：内部のコンテンツを右揃えにする
  - ＊ align-items-startクラス　　：内部のコンテンツを上揃えにする
  - ＊ align-items-centerクラス　　：内部のコンテンツを上下中央揃えにする
  - ＊ align-items-endクラス　　：内部のコンテンツを下揃えにする

157

第２章　Bootstrap 5.3.0 の利用

2.7　デザインレイアウト

2.7.2　ディスプレイ

・「bootstrap_sample_02_07_00.html」ファイルに下記コードを反映

```
  :
38  <div class="fs-5 text-danger my-5"><!--h2の範囲-->
39  <h2 class="fs-3 text-center border border-start-0 border-end-0 py-3 my-3">
      ネコ科の分類史
    </h2>
40  <div class="d-grid d-flex mb-3">
41  <div><img src="imgs/tiger-330.jpg"></div>
42  <div><img src="imgs/leopard-330.jpg"></div>
43  <div><img src="imgs/puma-330.jpg"></div>
44  </div>
  :
```

158

```
   :
45  <p class="mb-3 text-danger-emphasis">リンネの『自然の体系』で … </p>
46  <div class="d-flex justify-content-end">
47  <p class="text-danger-emphasis p-2 bg-danger-subtle text-center"
       style="padding-left: 40px!important; padding-right: 40px!important;">
      <small>出典: フリー百科事典『ウィキペディア（Wikipedia）』</small>
    </p>
48  </div>
49  </div><!--h2の範囲終了-->
   :
```


・**実行結果（Webブラウザで「bootstrap_sample_02_07_00.html」ファイルを表示）**

第2章　Bootstrap 5.3.0 の利用

2.7　デザインレイアウト

2.7.3　カード

・Bootstrapでは、画像、タイトル、文章、リスト、ボタンなどをまとめて表示するカードレイアウトが利用できる

［基本のレイアウト］

※画像、タイトル、文章、ボタン

［カードレイアウト］cardクラス

161

第2章　Bootstrap 5.3.0 の利用

2.7　デザインレイアウト

2.7.3　カード

・カードレイアウトは、ヘッダ部、ボディ部、フッタ部で構成される

※ヘッダ部とフッタ部は省略している

```
<div class="card p-2 bg-body-tertiary border-2">
  <div class="card-img-top">
    <img src="imgs/lion-330-card.jpg" class="w-100">
  </div>
  <div class="card-body text-center">
    <h3 class="card-title text-secondary-emphasis fs-5
               border-top border-bottom w-50 py-2 mx-auto
               text-center">ライオン</h3>
    <p class="card-text text-info-emphasis pb-2 mb-2">
      哺乳綱食肉目ネコ科ヒョウ属に分類される食肉類。
    </p>
    <button type="button" class="btn btn-primary mb-2">
      解説はこちら
    </button>
  </div>
</div>
```

162

第２章　Bootstrap 5.3.0 の利用

2.7　デザインレイアウト

2.7.3　カード

・「bootstrap_sample_02_07_00.html」ファイルに下記コードを反映

```
  :
50  <div><!--h2の範囲-->
51  <div class="container-fluid fs-5 text-danger my-5">
52  <h2 class="fs-3 text-center border border-start-0 border-end-0 py-3 my-3">
      ネコ科の肉食獣たち
    </h2>
53  <div class="mt-4 row">
  :
```


第２章　Bootstrap 5.3.0 の利用

2.7　デザインレイアウト

2.7.3　カード

```
  :
54  <div class="col-4 card mb-4 p-2 bg-body-tertiary border-2"><!--ライオンの範囲-->
55  <div class="card-img-top"><img src="imgs/lion-330-card.jpg" class="w-100"></div>
56  <div class="card-body text-center">
57  <h3 class="card-title text-secondary-emphasis fs-5 border-top border-bottom w-50 py-2
                mx-auto text-center">
      ライオン
    </h3>
58  <p class="card-text text-info-emphasis pb-2 mb-2">哺乳綱食肉目ネコ科ヒョウ … </p>
59  <button type="button" class="btn btn-primary mb-2">解説はこちら</button>
60  </div>
61  </div><!--ライオンの範囲終了-->
  :
```

```
  :
62  <div class="col-4 card mb-4 p-2 bg-body-tertiary border-2"><!--トラの範囲-->
63  <div class="card-img-top"><img src="imgs/tiger-330-card.jpg" class="w-100"></div>
64  <div class="card-body text-center">
65  <h3 class="card-title text-secondary-emphasis fs-5 border-top border-bottom w-50 py-2
            mx-auto text-center">
      トラ
    </h3>
66  <p class="card-text text-info-emphasis pb-2 mb-2">哺乳綱食肉目ネコ科ヒョウ … </p>
67  <button type="button" class="btn btn-primary mb-2">解説はこちら</button>
68  </div>
69  </div><!--トラの範囲終了-->
  :
```



```
  :
70  <div class="col-4 card mb-4 p-2 bg-body-tertiary border-2"><!--ヒョウの範囲-->
71  <div class="card-img-top"><img src="imgs/leopard-330-card.jpg" class="w-100"></div>
72  <div class="card-body text-center">
73  <h3 class="card-title text-secondary-emphasis fs-5 border-top border-bottom w-50 py-2
            mx-auto text-center">
      ヒョウ
    </h3>
74  <p class="card-text text-info-emphasis pb-2 mb-2">哺乳綱食肉目ネコ科ヒョウ … </p>
75  <button type="button" class="btn btn-primary mb-2">解説はこちら</button>
76  </div>
77  </div><!--ヒョウの範囲終了-->
  :
```

第２章　Bootstrap 5.3.0 の利用

2.7　デザインレイアウト

2.7.3　カード

```
   :
78  </div>
79  </div>
80  </div><!--h2の範囲終了-->
   :
```


第２章　Bootstrap 5.3.0 の利用

2.7　デザインレイアウト

2.7.3　カード

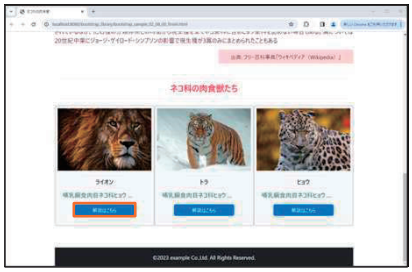
・**実行結果（Webブラウザで「bootstrap_sample_02_07_00.html」ファイルを表示）**

第2章　Bootstrap 5.3.0 の利用

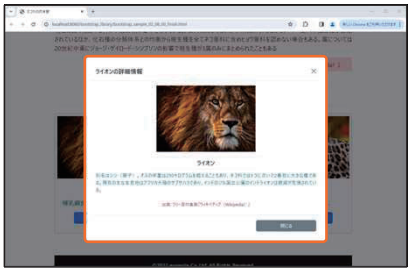
2.8　モーダル表示

2.8.1　モーダル

・モーダルは、Webページの最前面に表示され、それに対して何かしらの操作を行わない限り非表示にならない領域のこと



①ボタンをクリック



②モーダル表示

・BootstrapのCSSファイルがモーダルの装飾を行い、JavaScriptプログラムがモーダルの動作を制御する

169

第2章　Bootstrap 5.3.0 の利用

2.8　モーダル表示

2.8.1　モーダル

・モーダルの構築


```
<div class="modal fade" id="lion_id">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h1 class="modal-title">ライオンの詳細情報</h1>
        <button class="btn-close"></button>
      </div>
      <div class="modal-body">
        <img src="imgs/lion-330-card.jpg">
        <h3>ライオン</h3>
        <p>別名はシシ（獅子）。オスの体重は …</p>
        <p>出典: フリー百科事典 …</p>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn">閉じる</button>
      </div>
    </div>
  </div>
</div>
```

170

第2章　Bootstrap 5.3.0 の利用

2.8　モーダル表示

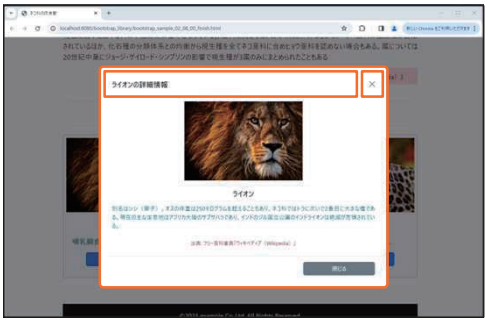
2.8.1　モーダル

・モーダルのオプション設定
- fadeクラス

  モーダルをフェードイン／フェードアウトで表示する
- modal-titleクラス

  モーダルのヘッダ部にタイトルを表示する
- btn-closeクラス

  モーダルのヘッダ部に「×ボタン」を表示する



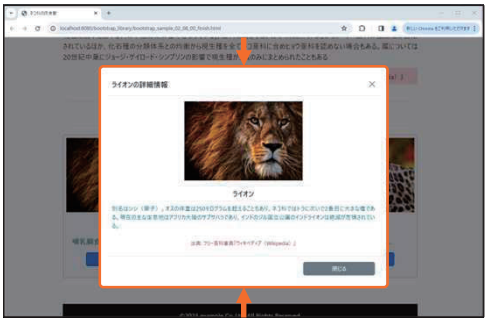
第2章　Bootstrap 5.3.0 の利用

2.8　モーダル表示

2.8.1　モーダル

・モーダルのオプション設定（続き）
- modal-dialog-centeredクラス

  モーダルを上下中央の位置に表示する



第2章　Bootstrap 5.3.0 の利用

2.8　モーダル表示

2.8.1　モーダル

・モーダルのオプション設定（続き）

- modal-smクラス

  モーダルをスモールサイズで表示する

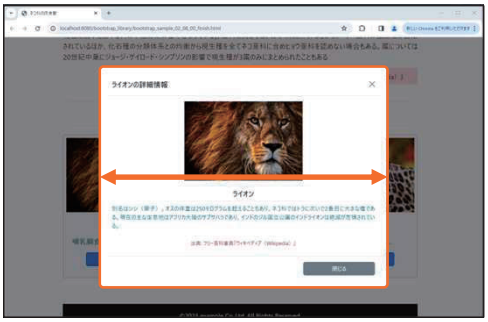
- modal-lgクラス

  モーダルをラージサイズで表示する

- modal-xlクラス

  モーダルをエクストララージサイズで表示する

※指定しない場合は、スモールとラージの中間サイズで表示される



※ラージサイズのモーダルダイアログ

第2章　Bootstrap 5.3.0 の利用

2.8　モーダル表示

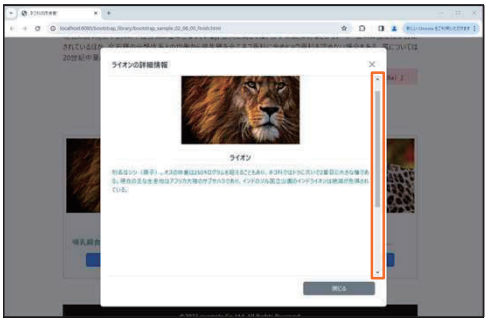
2.8.1　モーダル

・モーダルのオプション設定（続き）

- modal-dialog-scrollableクラス

  モーダルのコンテンツ部分に、自動的にスクロールバーを表示する

※指定しない場合は、Webページのグレーアウト部分に、自動的にスクロールバーが表示される



第2章　Bootstrap 5.3.0 の利用

2.8　モーダル表示

2.8.1　モーダル

・モーダルの表示設定

[モーダル]

※class="modal" id="lion_id" 設定済み

表示

[Webページ]


```
<button data-bs-toggle="modal"
        data-bs-target="#lion_id">
  解説はこちら
</button>
```


第2章　Bootstrap 5.3.0 の利用

2.8　モーダル表示

2.8.1　モーダル

・「bootstrap_sample_02_08_00.html」ファイルに下記コードを反映

```
   :
59  <button type="button"
          class="btn btn-primary mb-2" data-bs-toggle="modal" data-bs-target="#lion_id">
      解説はこちら
    </button>
   :
67  <button type="button"
          class="btn btn-primary mb-2" data-bs-toggle="modal" data-bs-target="#tiger_id">
      解説はこちら
    </button>
   :
```

第２章　Bootstrap 5.3.0 の利用

2.8　モーダル表示

2.8.1　モーダル

```
    :
75  <button type="button"
            class="btn btn-primary mb-2" data-bs-toggle="modal" data-bs-target="#leopard_id">
      解説はこちら
    </button>
    :
```


第２章　Bootstrap 5.3.0 の利用

2.8　モーダル表示

2.8.1　モーダル

```
    :
81  <!-- モーダルライオン本体 -->
82  <div class="modal fade" id="lion_id" data-bs-backdrop="static" data-bs-keyboard="false"
        tabindex="-1" aria-hidden="true" aria-labelledby="lion_1">
83  <div class="modal-dialog modal-lg modal-dialog-centered modal-dialog-scrollable text-center">
84  <div class="modal-content p-3">
85  <div class="modal-header">
86  <h1 class="modal-title fs-5" id="lion_1">ライオンの詳細情報</h1>
87  <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
88  </div>
    :
```

```
   :
89  <div class="modal-body">
90  <div class="mb-3"><img src="imgs/lion-330-card.jpg"></div>
91  <h3 class="text-secondary-emphasis fs-5 border-top border-bottom w-50 py-2 mx-auto
              text-center">
      ライオン
    </h3>
92  <p class="text-info-emphasis pb-2 mb-2">別名はシシ（獅子）。オスの体重は… </p>
93  <p class="text-danger-emphasis p-2 text-center">
      <small>出典: フリー百科事典『ウィキペディア（Wikipedia）』</small>
    </p>
94  </div>
   :
```



```
   :
95  <div class="modal-footer">
96  <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">閉じる</button>
97  </div>
98  </div><!-- modal-contentの終わり -->
99  </div><!-- modal-dialogの終わり -->
100 </div><!-- modalの終わり -->
101 <!-- モーダルライオン本体終了 -->
   :
```

```
    :
102  <!-- モーダルトラ本体 -->
103  <div class="modal fade" id="tiger_id" data-bs-backdrop="static" data-bs-keyboard="false"
         tabindex="-1" aria-hidden="true" aria-labelledby="tiger_1">
104  <div class="modal-dialog modal-lg modal-dialog-centered modal-dialog-scrollable text-center">
105  <div class="modal-content p-3">
106  <div class="modal-header">
107  <h1 class="modal-title fs-5" id="tiger_1">トラの詳細情報</h1>
108  <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
109  </div>
    :
```



```
    :
110  <div class="modal-body">
111  <div class="mb-3"><img src="imgs/tiger-330-card.jpg"></div>
112  <h3 class="text-secondary-emphasis fs-5 border-top border-bottom w-50 py-2 mx-auto
                text-center">
       トラ
     </h3>
113  <p class="text-info-emphasis pb-2 mb-2">トラ（虎、Panthera tigris）は、… </p>
114  <p class="text-danger-emphasis p-2 text-center">
       <small>出典: フリー百科事典『ウィキペディア（Wikipedia）』</small>
     </p>
115  </div>
    :
```

```
    :
116  <div class="modal-footer">
117  <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">閉じる</button>
118  </div>
119  </div><!-- modal-contentの終わり -->
120  </div><!-- modal-dialogの終わり -->
121  </div><!-- modalの終わり -->
122  <!-- モーダルトラ本体終了 -->
    :
```



```
    :
123  <!-- モーダルヒョウ本体 -->
124  <div class="modal fade" id="leopard_id" data-bs-backdrop="static" data-bs-keyboard="false"
         tabindex="-1" aria-hidden="true" aria-labelledby="leopard_1">
125  <div class="modal-dialog modal-lg modal-dialog-centered modal-dialog-scrollable text-center">
126  <div class="modal-content p-3">
127  <div class="modal-header">
128  <h1 class="modal-title fs-5" id="leopard_1">ヒョウの詳細情報</h1>
129  <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
130  </div>
    :
```

```
    :
131  <div class="modal-body">
132  <div class="mb-3"><img src="imgs/leopard-330-card.jpg"></div>
133  <h3 class="text-secondary-emphasis fs-5 border-top border-bottom w-50 py-2 mx-auto
                text-center">
       ヒョウ
     </h3>
134  <p class="text-info-emphasis pb-2 mb-2">アフリカ大陸からアラビア半島 … </p>
135  <p class="text-danger-emphasis p-2 text-center">
       <small>出典: フリー百科事典『ウィキペディア（Wikipedia）』</small>
     </p>
136  </div>
    :
```



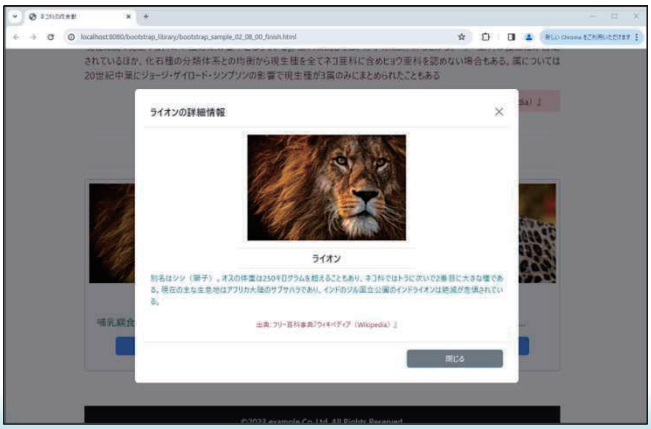
```
    :
137  <div class="modal-footer">
138  <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">閉じる</button>
139  </div>
140  </div><!-- modal-contentの終わり -->
141  </div><!-- modal-dialogの終わり -->
142  </div><!-- modalの終わり -->
143  <!-- モーダルヒョウ本体 -->
    :
```

第２章　Bootstrap 5.3.0 の利用

2.8　モーダル表示

2.8.1　モーダル

・**実行結果（Webブラウザで「bootstrap_sample_02_08_00.html」ファイルを表示 ）**



187

CSS/JavaScriptライブラリ
第３章
jQueryの概要

【本章学習内容】

本章では、jQueryの概要について学習します。

第３章　jQueryの概要

3.1　jQueryとは

3.1.1　jQueryの特長

- Webシステム開発で広く使われているJavaScriptのライブラリ
- jQueryを利用すると、JavaScriptプログラムを短時間で開発できる
- 複数のWebブラウザ上で生じる、JavaScriptプログラムの挙動の違いを抑えることができる
- GoogleマップやXの「いいね」の表示に使われている非同期通信「Ajax」を簡単に実現できる

第３章　jQueryの概要

3.1　jQueryとは

3.1.1　jQueryの特長

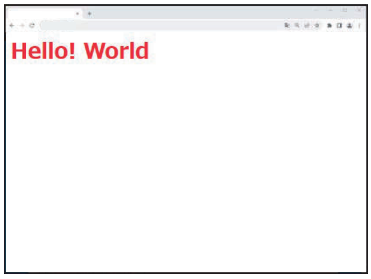
・DOM操作が簡単である

[JavaScriptのコード]（抜粋）

```
<h1 id="test">Hello! World</h1>
<script>
   'use strict';
   document.getElementById("test").style.color = "red";
</script>
```

[jQueryのコード]（抜粋）

```
<h1 id="test">Hello! World</h1>
<script>
   'use strict';
   $(function() {
      $("#test").css("color", "red");
   });
</script>
```



第３章　jQueryの概要

3.2　jQueryの利用方法

3.2.1　jQueryの入手

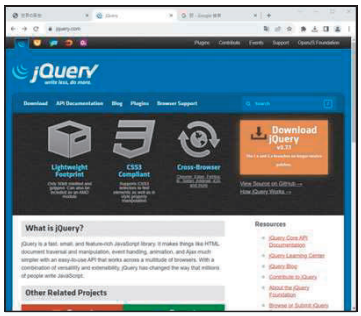
「jQuery公式サイト」より、 jQueryをダウンロードします。
インターネット接続環境をご準備ください。

[ダウンロード手順]

①Webブラウザを起動

②URL欄に「https://jquery.com/」を指定

③jQueryのWebサイトを表示



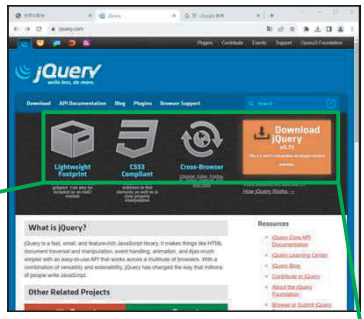
第３章　jQueryの概要

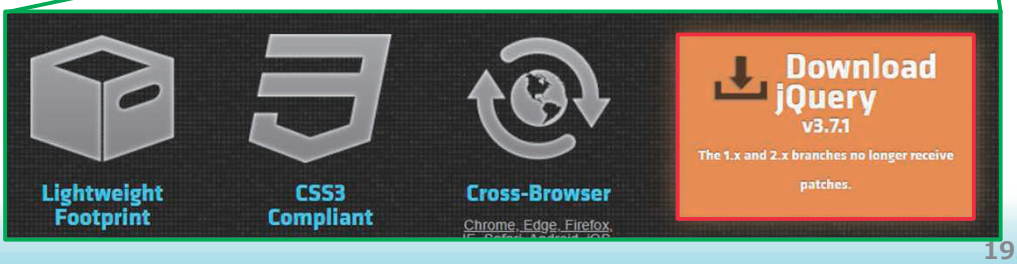
3.2　jQueryの利用方法

3.2.1　jQueryの入手

［ダウンロード手順］

④トップページの「ダウンロード」リンクをクリック





192

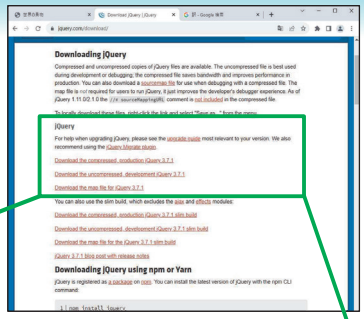
第３章　jQueryの概要

3.2　jQueryの利用方法

3.2.1　jQueryの入手

［ダウンロード手順］

⑤ダウンロードページの「Download the compressed, production jQuery 3.7.1」リンクを右クリックして「名前を付けてリンク先を保存…」を選択



193

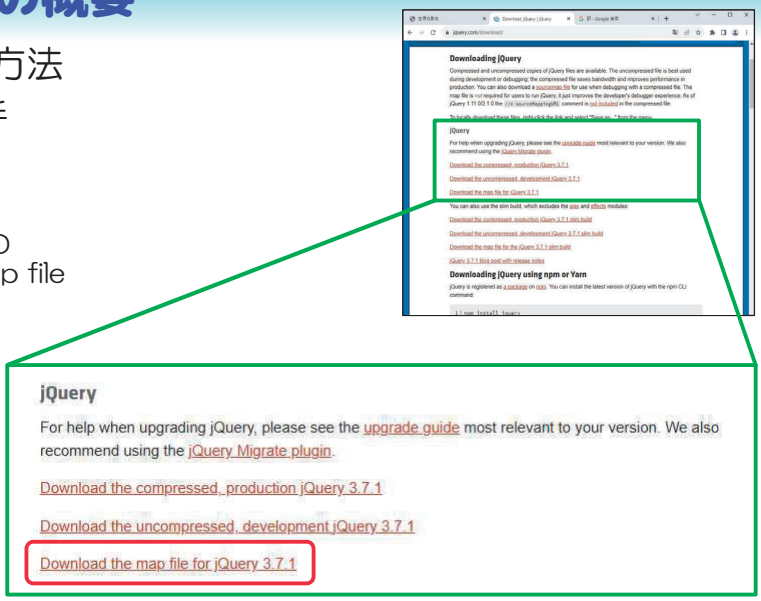
第３章　jQueryの概要

3.2　jQueryの利用方法

3.2.1　jQueryの入手

［ダウンロード手順］

⑥ダウンロードページの
「Download the map file
for jQuery 3.7.1」
リンクをクリック



第３章　jQueryの概要

3.2　jQueryの利用方法

3.2.1　jQueryの入手

［ダウンロード手順］

⑦次のファイルがあることを確認

- jquery-3.7.1.min.js
- jquery-3.7.1.min.map

第３章　jQueryの概要

3.2　jQueryの利用方法

3.2.1　jQueryの入手

［動的Webプロジェクトの準備］

①Eclipseを起動

②メニューの「ファイル」→「新規」→
「動的Webプロジェクト」を選択

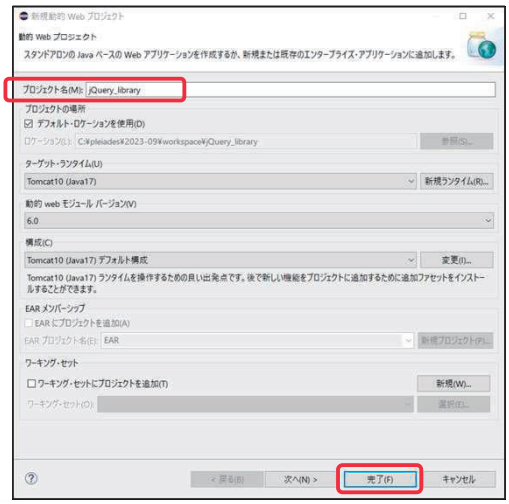
第３章　jQueryの概要

3.2　jQueryの利用方法

3.2.1　jQueryの入手

［動的Webプロジェクトの準備］

③プロジェクト名「jQuery_library」
を入力後、「完了」ボタンをクリック



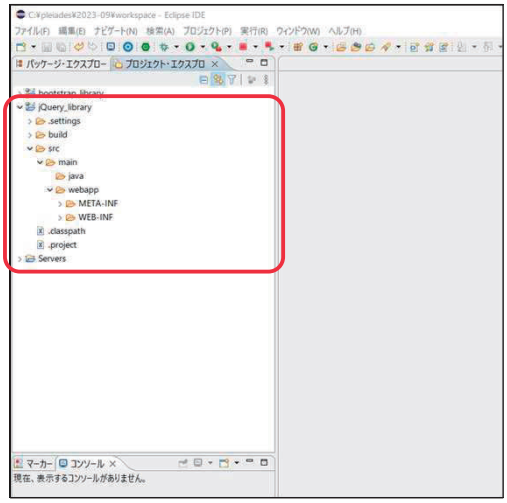
第３章　jQueryの概要

3.2　jQueryの利用方法

3.2.1　jQueryの入手

［動的Webプロジェクトの準備］

④「jQuery_library」→「src」→「main」→「webapp」の順番にクリックしフォルダを展開



198

第３章　jQueryの概要

3.2　jQueryの利用方法

3.2.1　jQueryの入手

［動的Webプロジェクトの準備］

⑤メニューの「ファイル」→「新規」→「HTMLファイル」を選択

199

第３章　jQueryの概要

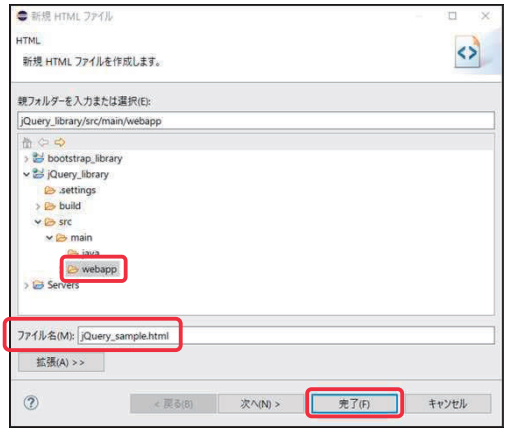
3.2　jQueryの利用方法

3.2.1　jQueryの入手

［動的Webプロジェクトの準備］

⑥保存フォルダ「webapp」を選択

⑦ファイル名「jQuery_sample.html」を入力し「完了」ボタンをクリック



200

第３章　jQueryの概要

3.2　jQueryの利用方法

3.2.2　CDN経由で利用

- CDN（Contents Delivery Network）は、大容量のWebコンテンツをインターネット経由で配信するためのネットワーク

  ［特長］

  - コンテンツ配信の高速化
  - 複数サーバによる冗長構成と負荷分散
  - 高度なセキュリティ対策
  - 運用コストの削減　など

- CDN経由でjQueryを利用するには

  【JavaScript】`<script src="https://code.jquery.com/jquery-3.7.1.min.js"></script>`

  を指定する

201

3.2　jQueryの利用方法

3.2.2　CDN経由で利用

・「jQuery_sample.html」ファイルに下記コードを追加

```
01  <!DOCTYPE html>
02  <html lang="ja">                    <!-- HTMLファイル内に日本語が存在する -->
03  <head>
04  <meta charset="UTF-8">
05  <title>Insert title here</title>
06  <script src="https://code.jquery.com/jquery-3.7.1.min.js"></script>
    :
```


3.2　jQueryの利用方法

3.2.2　CDN経由で利用

・「jQuery_sample.html」ファイルに下記コードを追加

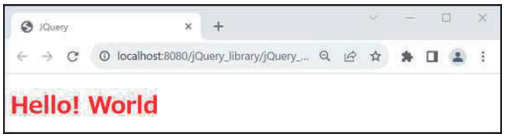
```
    :
07  <script>                                        <!-- jQueryのコード -->
08    $(function() {
09      $("#test").css("color", "red");          //h1タグの背景色を「赤」に設定する
10    });
11  </script>
12  </head>
13  <body>
14  <h1 id="test">Hello! world</h1>
15  </body>
16  </html>
```

第３章　jQueryの概要

3.2　jQueryの利用方法

3.2.2　CDN経由で利用

・実行結果（Webブラウザで「jQuery_sample.html」ファイルを表示）



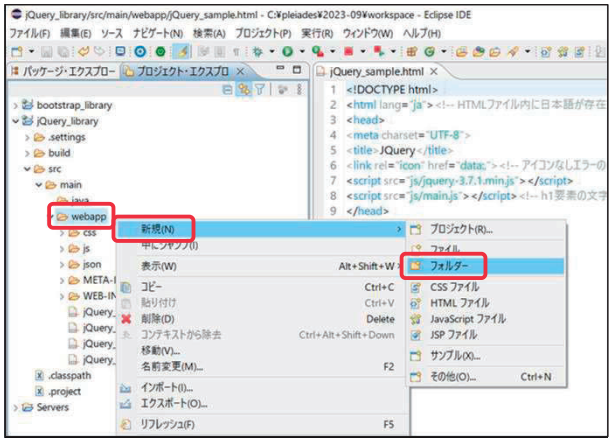
第３章　jQueryの概要

3.2　jQueryの利用方法

3.2.3　ローカルに配置して利用

［jQueryをローカルに配置］

①「webapp」フォルダを選択して、
メニューの「ファイル」→「新規」
→「フォルダー」を選択



第３章　jQueryの概要

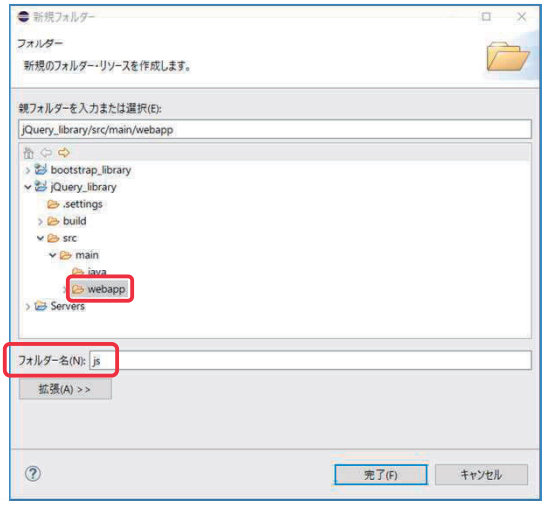
3.2　jQueryの利用方法

3.2.3　ローカルに配置して利用

［jQueryをローカルに配置］

②保存フォルダ「webapp」を選択

③フォルダ名「js」を入力し「完了」ボタンをクリック



206

第３章　jQueryの概要

3.2　jQueryの利用方法

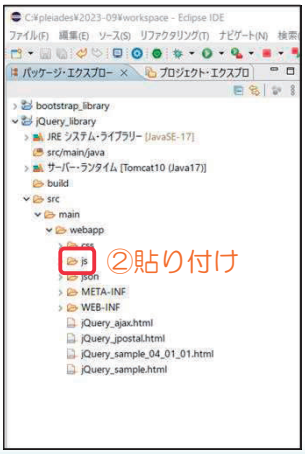
3.2.3　ローカルに配置して利用

［jQueryをローカルに配置］

④ダウンロードしたファイル
　jquery-3.7.1.min.js
　jquery-3.7.1.min.map
をコピーし、「js」フォルダに貼り付ける

エクスプローラ

Eclipse



207

第３章　jQueryの概要

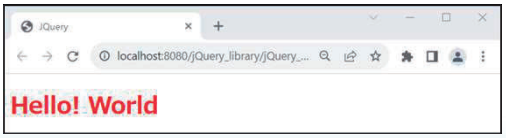
3.2　jQueryの利用方法

3.2.3　ローカルに配置して利用

・「jQuery_sample.html」ファイルに下記コードを追加

```
   :
05  <title>Insert title here</title>
06  <script src="js/jquery-3.7.1.min.js"></script>
07  <script>
   :
```

・**実行結果（Webブラウザで「jQuery_sample.html」ファイルを表示）**



第３章　jQueryの概要

3.3　非同期通信とは

3.3.1　同期通信と非同期通信

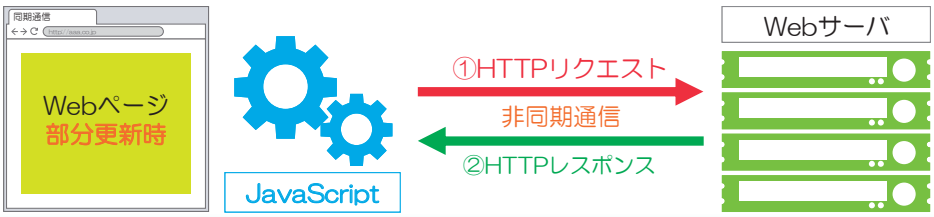
・同期通信
- HTTPレスポンスの受信が完了するまでの間、Webブラウザは他の処理を行わず待機を続ける通信方式
- WebブラウザとWebサーバの間の「通常の通信方式」
- Webページの遷移時に利用

第3章　jQueryの概要

3.3　非同期通信とは

3.3.1　同期通信と非同期通信

- 非同期通信
  - HTTPレスポンスの受信が完了するまでの間、Webブラウザは他の処理を行うことができる通信方式
  - JavaScriptプログラムとWebサーバの間で利用される「特別な通信方式」
  - Webページの部分更新時に利用



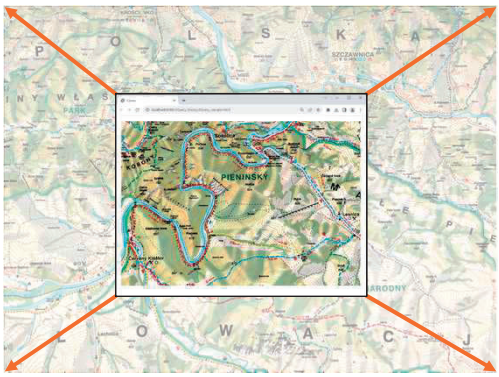
210

第3章　jQueryの概要

3.3　非同期通信とは

3.3.2　JavaScriptの非同期通信「Ajax」

- 非同期通信「Ajax」
  （Asynchronous JavaScript + XML）
- JavaScriptの拡張機能
- 非同期通信と同期通信の機能を持っているが、同期通信が選択されるケースはほとんどない
- Googleマップの表示やXの「いいね」の表示などで使われている
- 非同期通信はjQueryの「$.ajaxメソッド」によって簡単に実現できる



地図はドラッグしても
すぐに表示される

211

第３章　jQueryの概要

3.3　非同期通信とは

3.3.2　JavaScriptの非同期通信「Ajax」

［学習用教材の準備］

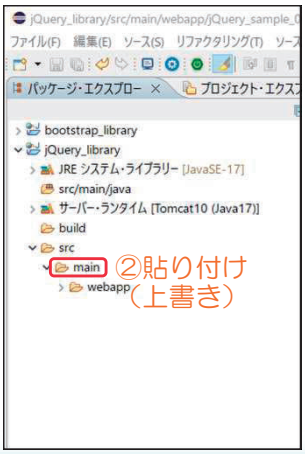
①第３章の配布プログラム「Library第３章.zip」を展開（解凍）

②「webapp」フォルダをコピー

③Eclipseで「jQuery_library」→「src」と順番に展開し、「main」フォルダ上で貼り付け

エクスプローラ

Eclipse



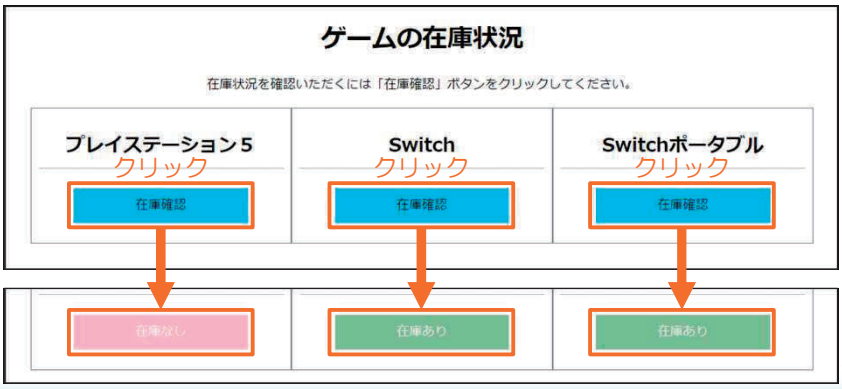
212

第３章　jQueryの概要

3.3　非同期通信とは

3.3.2　JavaScriptの非同期通信「Ajax」

・Ajaxサンプル「ゲームの在庫状況」（jQuery_ajax.htmlを「サーバで実行」する）



213

CSS/JavaScriptライブラリ
第４章
jQuery 3.7.0 の利用

【本章学習内容】

本章では、jQuery 3.7.0 の利用方法について学習します。

214

第４章　jQuery 3.7.0 の利用

4.1　jQueryプラグイン

4.1.1　jquery.jpostal.js

［学習用教材の準備］

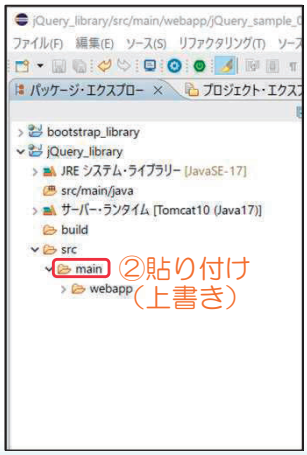
①第４章の配布プログラム「Library第４章.zip」を展開（解凍）

②「webapp」フォルダをコピー

③Eclipseで「jQuery_library」→「src」と順番に展開し、「main」フォルダ上で貼り付け

エクスプローラ

Eclipse



215

第４章　jQuery 3.7.0 の利用

４.１　jQueryプラグイン

４.１.１　jquery.jpostal.js

・郵便番号を入力すると、住所が自動的に表示されるプラグイン

【ダウンロードURL】https://github.com/ninton/jquery.jpostal.js/
※第４章の配布プログラムに含まれているので、本講座ではダウンロードしなくてよい。

第４章　jQuery 3.7.0 の利用

４.１　jQueryプラグイン

４.１.１　jquery.jpostal.js

・「plugin_sample.js」ファイルに下記コードを反映

```
:
01  $(function() {
02    $('#post').jpostal({
03      postcode : [ '#post' ],
04      address : { '#address1' : '%3%4%5' },
05      url : { 'http' : 'json/' }
06    });
07  });
```

%3 … 都道府県名
%4 … 市区町村名
%5 … 町名

郵便番号と住所を格納したJSONファイルの保存場所

第４章　jQuery 3.7.0 の利用

４.１　jQueryプラグイン

４.１.１　jquery.jpostal.js

・「jQuery_sample_04_01_00.html」ファイルに下記コードを反映

```
  :
07  <script src="js/jquery-3.7.1.min.js"></script>
08  <script src="js/jquery.jpostal.js"></script>
09  <script src="js/plugin_sample.js"></script>
10  <link href="css/bootstrap.min.css" rel="stylesheet">
11  <link href="css/style.css" rel="stylesheet">
  :
```


第４章　jQuery 3.7.0 の利用

４.１　jQueryプラグイン

４.１.１　jquery.jpostal.js

```
  :
23  <div class="mb-3">
24  <label for="post" class="col-form-label">郵便番号</label>
25  <div class="input-group">
26  <input type="text" id="post" name="post" class="form-control" placeholder="000-0000で入力">
27  </div>
28  </div>
  :
```

第４章　jQuery 3.7.0 の利用

４.１　jQueryプラグイン

４.１.１　jquery.jpostal.js

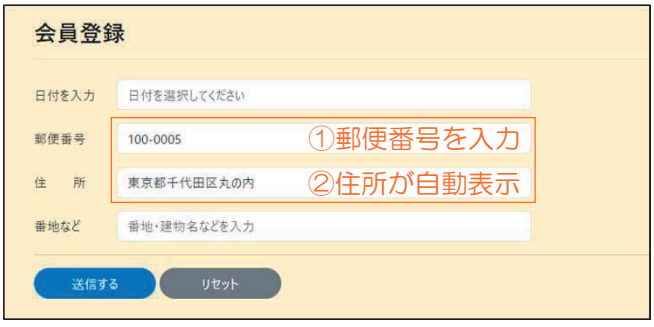
```
   :
29  <div class="mb-3">
30  <label for="address1" class="col-form-label">住　　所</label>
31  <div class="input-group">
32  <input type="text" id="address1" name="address1" class="form-control"
        placeholder="郵便番号が入力されると住所は自動で表示されます">
33  </div>
34  </div>
   :
```


第４章　jQuery 3.7.0 の利用

４.１　jQueryプラグイン

４.１.１　jquery.jpostal.js

・**実行結果（Webブラウザで「jQuery_sample_04_01_00.html」ファイルを表示）**

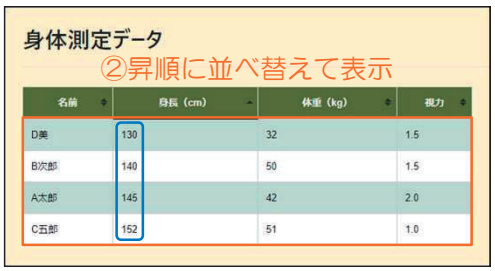


第４章　jQuery 3.7.0 の利用

４.１　jQueryプラグイン

４.１.２　jquery.tablesorter.js

・テーブルの、ヘッダ部のマークをクリックすると、データ部が昇順または降順に並べ替えて表示されるプラグイン



【ダウンロードURL】https://github.com/Mottie/tablesorter/

※第４章の配布プログラムに含まれているので、本講座ではダウンロードしなくてよい。

222

第４章　jQuery 3.7.0 の利用

４.１　jQueryプラグイン

４.１.２　jquery.tablesorter.js

・「plugin_sample.js」ファイルに下記コードを反映

```
：
09  $(function() {
10    $('#p_measurement').tablesorter({
11      headers: {
12        0: { sorter: 'text'},          //文字列として並べ替える
13        1: { sorter: 'digit'},         //数値として並べ替える
14        2: { sorter: 'digit'},         //数値として並べ替える
15        3: { sorter: 'digit'}          //数値として並べ替える
16      }
17    });
18  });
```

223

第４章　jQuery 3.7.0 の利用

４.１　jQueryプラグイン

４.１.２　jquery.tablesorter.js

・「jQuery_sample_04_01_00.html」ファイルに下記コードを反映

```
  :
07  <script src="js/jquery-3.7.1.min.js"></script>
08  <script src="js/jquery.jpostal.js"></script>
09  <script src="js/jquery.tablesorter.min.js"></script>
10  <script src="js/plugin_sample.js"></script>
11  <link href="css/theme.default.min.css" rel="stylesheet">
12  <link href="css/bootstrap.min.css" rel="stylesheet">
13  <link href="css/style.css" rel="stylesheet">
  :
```


第４章　jQuery 3.7.0 の利用

４.１　jQueryプラグイン

４.１.２　jquery.tablesorter.js

・「jQuery_sample_04_01_00.html」ファイルに下記コードを反映

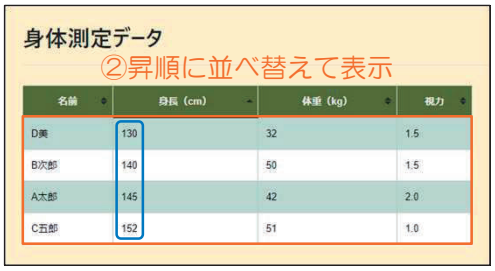
```
  :
47  <h2 class="fs-3 mb-4 pb-3 border-bottom">身体測定データ</h2>
48  <table id="p_measurement" class="w-75 table table-bordered table-striped">
49  <thead>
  :
56  </thead>
  :
83  </table>
  :
```

第４章　jQuery 3.7.0 の利用

４.１　jQueryプラグイン

４.１.２　jquery.tablesorter.js

・**実行結果（Webブラウザで「jQuery_sample_04_01_00.html」ファイルを表示）**



226

第４章　jQuery 3.7.0 の利用

４.１　jQueryプラグイン

４.１.３　jquery-ui.css、jquery-ui.js、datepicker-ja.js

・Webページ上に日付を選択するためのカレンダーが表示されるプラグイン

【ダウンロードURL】https://jqueryui.com/download/

※第４章の配布プログラムに含まれているので、本講座ではダウンロードしなくてよい。

227

第４章　jQuery 3.7.0 の利用

４.１　jQueryプラグイン

４.１.３　jquery-ui.css、jquery-ui.js、datepicker-ja.js

・「plugin_sample.js」ファイルに下記コードを反映

```
：
$(function() {
    $('#datepicker').datepicker({ dateFormat : 'yy年m月d日' });
});
```


第４章　jQuery 3.7.0 の利用

４.１　jQueryプラグイン

４.１.３　jquery-ui.css、jquery-ui.js、datepicker-ja.js

・「jQuery_sample_04_01_00.html」ファイルに下記コードを反映

```
：
<script src="js/jquery-3.7.1.min.js"></script>
<script src="js/jquery.jpostal.js"></script>
<script src="js/jquery.tablesorter.min.js"></script>
<script src="js/jquery-ui.min.js"></script>
<script src="js/datepicker-ja.js"></script>
<script src="js/plugin_sample.js"></script>
：
```

```
    :
13  <link href="css/theme.default.min.css" rel="stylesheet">
14  <link href="css/jquery-ui.min.css" rel="stylesheet">
15  <link href="css/bootstrap.min.css" rel="stylesheet">
16  <link href="css/style.css" rel="stylesheet">
    :
```


第４章　jQuery 3.7.0 の利用
４.１　jQueryプラグイン
４.１.３　jquery-ui.css、jquery-ui.js、datepicker-ja.js

```
    :
22  <div class="mb-3">
23  <label for="date" class="col-form-label">日付を入力</label>
24  <div class="input-group">
25  <input type="text" id="datepicker" name="date" class="form-control"
        placeholder="日付を選択してください">
26  </div>
27  </div>
    :
```

第４章　jQuery 3.7.0 の利用

４.１　jQueryプラグイン

４.１.３　jquery-ui.css、jquery-ui.js、datepicker-ja.js

・**実行結果（Webブラウザで「jQuery_sample_04_01_00.html」ファイルを表示）**

232

第２章　プログラミング演習

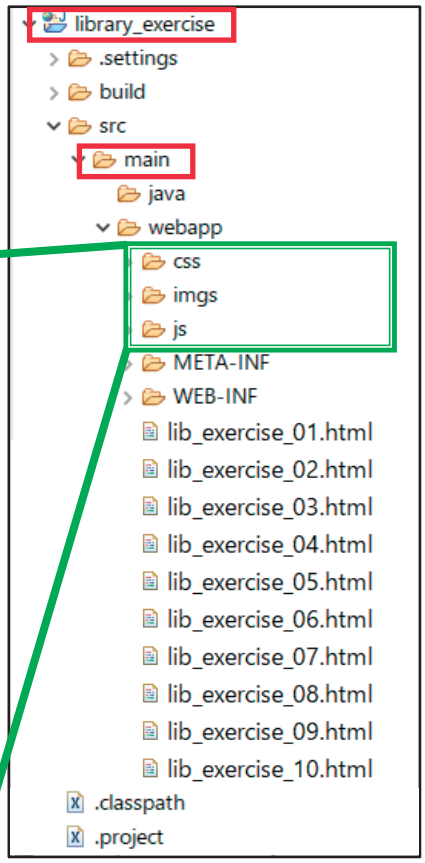
事前準備

①Eclipse上で動的Webプロジェクトを作成する。
　　プロジェクト名：library_exercise
②配布データをコピーする
　　配布フォルダ名：webapp
③配布データを貼り付ける（上書き）
　　貼付先　　：main

CSSフォルダ
- bootstrap.min.css
- bootstrap.min.css.map
- style.css
- style_pure.css

imgsフォルダ
- cats-abyssinians.png
- cats-bengal.png
- cats-british-shorthair.png
- cats-european-shorthair.png
- cats-european-shorthair2.png
- cats-scothishfold.png
- dogs-beegle.png
- dogs-bulldog.png
- dogs-golden.png
- dogs-kai.png
- dogs-rub.png
- golden-eagle-top.jpg
- golden-eagle_1_1000.jpg
- golden-eagle_1_330.jpg
- golden-eagle_2_1000.jpg
- golden-eagle_2_330.jpg
- golden-eagle_3_1000.jpg
- golden-eagle_3_330.jpg
- golden-eagle_4_1000.jpg
- golden-eagle_4_330.jpg
- golden_eagle_lifestyle.jpg
- hawk_330.jpg
- logo.png
- pet_food.png
- pet_toys.png
- top_img.png

jsフォルダ
- bootstrap.min.js
- bootstrap.min.js.map
- validation.js



次の**問題01**～**問題10**を読んで、提示された**[要件]**と**[実行結果]**を満たすプログラムを作成しなさい。

問題01（難易度 ☆）

・「イヌワシの生態」を解説するWebページを作成する。
・「lib_exercise_01.html」ファイルを開き、HTMLプログラムを入力する
・プログラム作成後に「lib_exercise_01.html」ファイルを実行し、Webブラウザに表示される実行結果を確認する

[要件]

・WebページにBootstrap 5.3.0を適用し、Webページのデザイン性を高める

[設定内容]

・「bootstrap.min.css」ファイルを読み込む
・「メイン」コンテンツの幅に親要素の75%を設定する
・「イヌワシの生態」の画像と解説のレイアウトに「d-flex」クラスを適用する
・見出し「イヌワシの生態」の下部に境界線を設定する
・見出し「イヌワシの生態」の文字色に「text-danger」クラスを適用する
・小見出し「Golden Eagle Ecology」の文字色に「text-warning」クラスを適用する
・「猛禽類の定義の変遷」の画像と解説のレイアウトに「d-flex」クラスを適用する
・「猛禽類の定義の変遷」の画像と解説の背景色に「bg-light」クラスを適用する
・見出し「猛禽類の定義の変遷」の下部に境界線を表示する
・見出し「猛禽類の定義の変遷」の文字色に「text-dark」クラスを適用する
・小見出し「Birds Of Prey」の文字色に「text-warning」クラスを適用する
・「bootstrap.min.js」ファイルを読み込む

[実行結果]

lib_exercise_01.html

```
01  <!DOCTYPE html>
02  <html>
03  <head>
04    <meta charset="UTF-8">
05    <title>Golden Eagle</title>
06    <link href="                    " rel="stylesheet">
07    <link href="css/style.css" rel="stylesheet">
08  </head>
09  <body>
10    <main class="    mt-5">
11      <article>
12        <div class="      ">
13          <div class="me-4">
14            <img src="imgs/golden_eagle_lifestyle.jpg" alt="">
15          </div>
16          <div>
17            <h1 class="fs-2                mb-4 pb-3           ">
                イヌワシの生態<br>
                <span class="fw-normal fs-5           ">Golden Eagle Ecology</span>
              </h1>
18            <p class="mb-4">全長 75 - 95 センチメートル。… </p>
19            <p class="px-3 py-2 bg-warning-subtle d-inline">
                <small>出典: フリー百科事典『ウィキペディア (Wikipedia) 』</small>
              </p>
20          </div>
21        </div>
22      </article>
23      <article class="pt-5">
24        <div class="      mt-5 border p-4        ">
25          <div>
26            <h2 class="fs-2                mb-4 pb-2           ">
                猛禽類の定義の変遷<br>
                <span class="fw-normal fs-5           ">Birds Of Prey</span>
              </h2>
27            <p class="mb-4">ワシ、タカ、ハゲワシ、ハヤブサ、コンドル、フクロウが … </p>
28            <p class="px-3 py-2 bg-warning-subtle d-inline">
                <small>出典: フリー百科事典『ウィキペディア (Wikipedia) 』</small>
              </p>
29          </div>
30          <div class="ms-4">
31            <img src="imgs/hawk_330.jpg" alt="">
32          </div>
33        </div>
34      </article>
35    </main>
36    <script src="                    "></script>
37  </body>
38  </html>
```

問題02（難易度 ☆）

・「イヌワシの生態」を解説するWebページを作成する。
・「lib_exercise_02.html」ファイルを開き、HTMLプログラムを入力する
・プログラム作成後に「lib_exercise_02.html」ファイルを実行し、Webブラウザに表示される実行結果を確認する

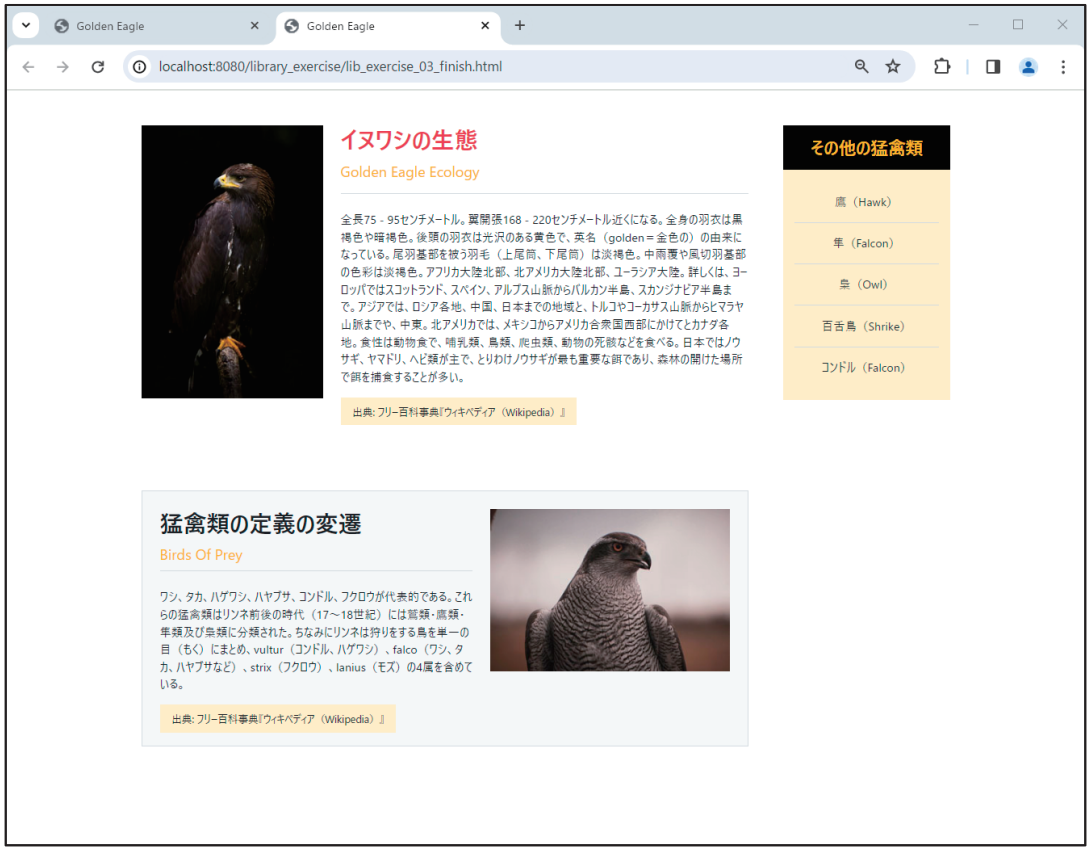
［要件］

・WebページにBootstrap 5.3.0を適用し、Webページのデザイン性を高める

［設定内容］

・コンテンツ全体の幅に親要素の75%を設定する
・コンテンツ全体のレイアウトに「d-flex」クラスを適用する
・コンテンツ全体の外側の左右余白に自動を設定する
・「サイド」コンテンツの幅に親要素の25%を設定する
・見出し「その他の猛禽類」の背景色に「bg-black」クラスを適用する
・見出し「その他の猛禽類」の文字列に左右中央揃えを設定する
・見出し「その他の猛禽類」の文字色に「text-warning」クラスを適用する
・順序なしリストの背景色に「bg-warning-subtle」クラスを適用する
・順序なしリストの文字列に左右中央揃えを設定する
・順序なしリストの文字色に「text-secondary-emphasis」クラスを適用する
・「鷹（Hawk）」項目、「隼（Falcon）」項目、「梟（Owl）」項目、「百舌鳥（Shrike）」項目、「コンドル（Condor）」項目のリンクについて、レイアウトに「d-block」クラスを適用する

［実行結果］



lib_exercise_02.html

```
01  <!DOCTYPE html>
02  <html>
03  <head>
04    <meta charset="UTF-8">
05    <title>Golden Eagle</title>
06    <link href="css/bootstrap.min.css" rel="stylesheet">
07    <link href="css/style.css" rel="stylesheet">
08  </head>
09  <body>
10    <div class="                  ">
11      <main class="w-75 mt-5">
      :
36      </main>
37      <aside class="     mt-5 ps-5">
38        <h2 class="fs-4          mb-0 p-3                    ">その他の猛禽類</h2>
39        <ul style="list-style-type: none; padding-left: 0;"
              class="                p-3                                ">
40          <li class="p-3 border-bottom"><a class="          " href="#">鷹 (Hawk) </a></li>
41          <li class="p-3 border-bottom"><a class="          " href="#">隼 (Falcon) </a></li>
42          <li class="p-3 border-bottom"><a class="          " href="#">梟 (Owl) </a></li>
43          <li class="p-3 border-bottom"><a class="          " href="#">百舌鳥 (Shrike) </a></li>
44          <li class="p-3"><a class="          " href="#">コンドル (Condor) </a></li>
45        </ul>
46      </aside>
47    </div>
48    <script src="js/bootstrap.min.js"></script>
49  </body>
50  </html>
```

問題03（難易度 ☆）

・「イヌワシの生態」を解説するWebページを作成する。
・「lib_exercise_03.html」ファイルを開き、HTMLプログラムを入力する
・プログラム作成後に「lib_exercise_03.html」ファイルを実行し、Webブラウザに表示される実行結果を確認する

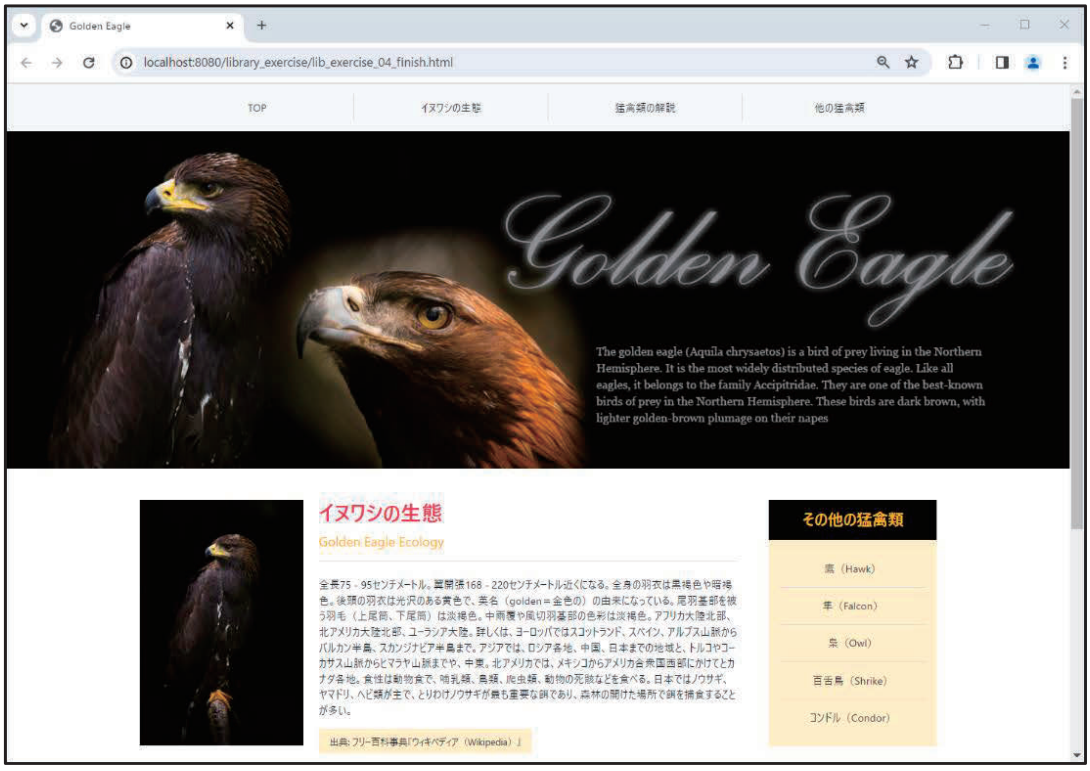
［要件］

・WebページにBootstrap 5.3.0を適用し、Webページのデザイン性を高める

［設定内容］

・ヘッダ領域をWebページの上部に固定する
・ヘッダ領域の幅に親要素の100%を設定する
・ヘッダ領域の背景色に「bg-light」クラスを適用する
・順序なしリストの幅に親要素の75%を設定する
・順序なしリストの外側の左右余白に自動を設定する
・順序なしリストのレイアウトに「d-flex」クラスを適用する
・順序なしリストの文字列に左右中央揃えを設定する
・順序なしリストの文字列の文字色に「text-secondary-emphasis」クラスを適用する
・「Top」項目、「イヌワシの生態」項目、「猛禽類の解説」項目、「他の猛禽類」項目の幅に親要素の25%を設定する
・バナー画像の領域の背景色に「bg-black」クラスを適用する
・バナー画像の幅に親要素の100%を設定する
・フッタ領域の背景色に「bg-dark」クラスを適用する
・フッタの文字色に「text-light」クラスを適用する
・フッタ内の段落の文字列に左右中央揃えを設定する

［実行結果］



lib_exercise_03.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Golden Eagle</title>
  <link href="css/bootstrap.min.css" rel="stylesheet">
  <link href="css/style.css" rel="stylesheet">
</head>
<body>
  <div class="                                    ">
    <header>
      <nav>
        <ul style="list-style-type: none; padding-left: 0;"
            class="                                    pt-3                                    ">
          <li class="      p-2 border-end"><a class="d-block" href="#">TOP</a></li>
          <li class="      p-2 border-end"><a class="d-block" href="#">イヌワシの生態</a></li>
          <li class="      p-2 border-end"><a class="d-block" href="#">猛禽類の解説</a></li>
          <li class="      p-2"><a class="d-block" href="#">他の猛禽類</a></li>
        </ul>
      </nav>
    </header>
  </div>
  <div class="          " style="padding-top: 80px;">
    <img src="imgs/golden-eagle-top.jpg" alt="" class="        ">
  </div>
  <div class="d-flex w-75 mx-auto">
    <main class="w-75 mt-5">
   :
    </main>
    <aside class="w-25 mt-5 ps-5">
   :
    </aside>
  </div>
  <div class="          mt-5 p-3">
    <footer class="fs-5              ">
      <p class="                ">
        <small>&copy;2023 example Co.,Ltd. All Rights Reserved.</small>
      </p>
    </footer>
  </div>
  <script src="js/bootstrap.min.js"></script>
</body>
</html>
```

問題04（難易度 ☆☆）

・「イヌワシの生態」を解説するWebページを作成する。
・「lib_exercise_04.html」ファイルを開き、HTMLプログラムを入力する
・プログラム作成後に「lib_exercise_04.html」ファイルを実行し、Webブラウザに表示される実行結果を確認する

［要件］

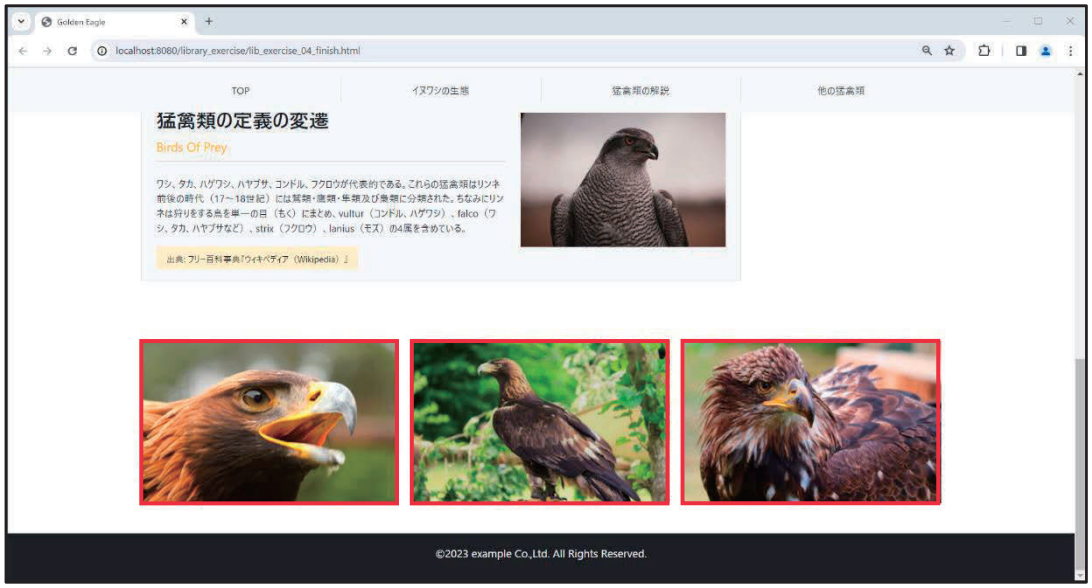
・WebページにBootstrap 5.3.0を適用し、Webページのデザイン性を高める
・Webページ内の「３枚のイヌワシ画像」をクリックすると、その画像がモーダル表示される

［設定内容］

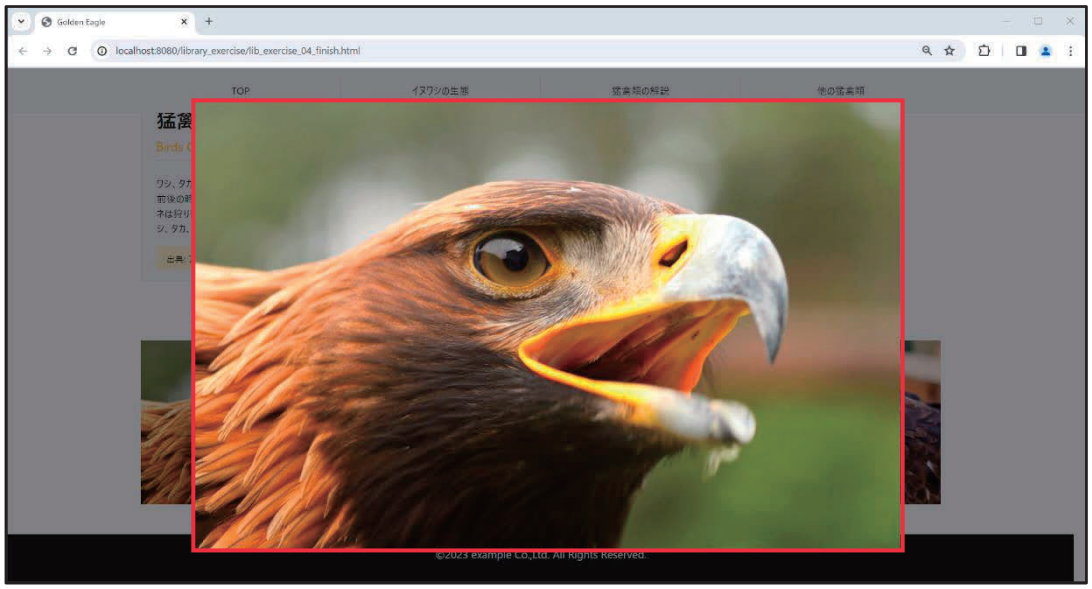
・３枚のイヌワシ画像の領域について、レイアウトに「d-flex」クラスを適用する
・３枚のイヌワシ画像の領域について、幅に親要素の75%を設定する
・３枚のイヌワシ画像の領域について、外側の左右余白に自動を設定する
・左側のイヌワシ画像について、幅に親要素の100%を設定する
・左側のイヌワシ画像にdata-bs-toggle属性を追加して「modal」を指定する
・左側のイヌワシ画像にdata-bs-target属性を追加して「#modal_id_1」を指定する
・中央のイヌワシ画像について、幅に親要素の100%を設定する
・中央のイヌワシ画像にdata-bs-toggle属性を追加して「modal」を指定する
・中央のイヌワシ画像にdata-bs-target属性を追加して「#modal_id_2」を指定する
・右側のイヌワシ画像について、幅に親要素の100%を設定する
・右側のイヌワシ画像にdata-bs-toggle属性を追加して「modal」を指定する
・右側のイヌワシ画像にdata-bs-target属性を追加して「#modal_id_3」を指定する
・左側のイヌワシ画像のモーダルに「modal」クラスを適用する
・左側のイヌワシ画像のモーダルについて、表示方法にフェードを設定する
・左側のイヌワシ画像のモーダルにモーダルダイアログを設定する
・左側のイヌワシ画像のモーダルについて、表示位置に上下中央揃えを設定する
・左側のイヌワシ画像のモーダルについて、表示サイズにエクストララージを設定する
・左側のイヌワシ画像のモーダル内の画像について、幅に親要素の100%を設定する
・中央のイヌワシ画像のモーダルに「modal」クラスを適用する
・中央のイヌワシ画像のモーダルについて、表示方法にフェードを設定する
・中央のイヌワシ画像のモーダルにモーダルダイアログを設定する
・中央のイヌワシ画像のモーダルについて、表示位置に上下中央揃えを設定する
・中央のイヌワシ画像のモーダルについて、表示サイズにエクストララージを設定する
・中央のイヌワシ画像のモーダル内の画像について、幅に親要素の100%を設定する
・右側のイヌワシ画像のモーダルに「modal」クラスを適用する
・右側のイヌワシ画像のモーダルについて、表示方法にフェードを設定する
・右側のイヌワシ画像のモーダルにモーダルダイアログを設定する
・右側のイヌワシ画像のモーダルについて、表示位置に上下中央揃えを設定する
・右側のイヌワシ画像のモーダルについて、表示サイズにエクストララージを設定する
・右側のイヌワシ画像のモーダル内の画像について、幅に親要素の100%を設定する

［実行結果］

（通常表示）



（画像をクリックしてモーダル表示）



lib_exercise_04.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Golden Eagle</title>
  <link href="css/bootstrap.min.css" rel="stylesheet">
  <link href="css/style.css" rel="stylesheet">
</head>
<body>
   :
  <div class="d-flex w-75 mx-auto">
   :
  </div>
  <div class="                    mt-5 pt-5">
    <div class="me-4" style="width: 33.3%;">
      <img class="          " src="imgs/golden-eagle_1_330.jpg" alt="" data-bs-toggle="          "
           data-bs-target="#          ">
    </div>
    <div class="me-4" style="width: 33.3%;">
      <img class="          " src="imgs/golden-eagle_2_330.jpg" alt="" data-bs-toggle="          "
           data-bs-target="#          ">
    </div>
    <div class="me-4" style="width: 33.3%;">
      <img class="          " src="imgs/golden-eagle_3_330.jpg" alt="" data-bs-toggle="          "
           data-bs-target="#          ">
    </div>
  </div>
  <div class="bg-dark mt-5 p-3">
   :
  </div>
  <div class="                    " id="modal_id_1" tabindex="-1" aria-hidden="true">
    <div class="                                                  ">
      <img class="          " src="imgs/golden-eagle_1_1000.jpg" alt="">
    </div>
  </div>
  <div class="                    " id="modal_id_2" tabindex="-1" aria-hidden="true">
    <div class="                                                  ">
      <img class="          " src="imgs/golden-eagle_2_1000.jpg" alt="">
    </div>
  </div>
  <div class="                    " id="modal_id_3" tabindex="-1" aria-hidden="true">
    <div class="                                                  ">
      <img class="          " src="imgs/golden-eagle_3_1000.jpg" alt="">
    </div>
  </div>
  <script src="js/bootstrap.min.js"></script>
</body>
</html>
```

問題05（難易度 ☆☆）

・「ペットショップPure」を紹介するWebページを作成する。
・「lib_exercise_05.html」ファイルを開き、HTMLプログラムを入力する
・プログラム作成後に「lib_exercise_05.html」ファイルを実行し、Webブラウザに表示される実行結果を確認する

［要件］

・WebページにBootstrap 5.3.0を適用し、Webページのデザイン性を高める

［設定内容］

・「bootstrap.min.css」ファイルを読み込む
・「当店のご紹介」の領域について、レイアウトに「container」クラスを適用する
・「当店のご紹介」の段落について、文字列に左右中央揃えを設定する
・３つの特長の領域に行を設定する
・３つの特長の領域について、幅に親要素の75%を設定する
・３つの特長の領域について、外側の左右余白に自動を設定する
・「優良ブリーダーとの取引」の領域に４列分を設定する
・「優良ブリーダーとの取引」の説明文の段落について、文字列に左揃えを設定する
・「優良ブリーダーとの取引」の「詳細はこちら」段落について、形状に「両端の角を丸める」に設定する
・「ペットの健康管理」の領域に４列分を設定する
・「ペットの健康管理」の説明文の段落について、文字列に左揃えを設定する
・「ペットの健康管理」の「詳細はこちら」段落について、形状に「両端の角を丸める」に設定する
・「店舗での取り組み」の領域に４列分を設定する
・「店舗での取り組み」の説明文の段落について、文字列に左揃えを設定する
・「店舗での取り組み」の「詳細はこちら」段落について、形状に「両端の角を丸める」を設定する
・「bootstrap.min.js」ファイルを読み込む

［実行結果］

lib_exercise_05.html

```
01 <!DOCTYPE html>
02 <html lang="ja">
03 <head>
04   <meta charset="utf-8">
05   <meta name="viewport" content="width=device-width, initial-scale=1">
06   <link href="                    " rel="stylesheet" type="text/css">
07   <link href="css/style_pure.css" rel="stylesheet" type="text/css">
08   <title>ペットショップPure（ピュア）</title>
09 </head>
10 <body>
11   <div class="mx-auto w-75">
12     <main>
13       <article>
14         <div class="          "          mb-5" id="info" style="padding-top: 150px;">
15           <h2 class="fw-normal p-2 text-primary-emphasis w-50 mx-auto border-top
                        border-bottom border-primary-subtle border-2">
             当店のご紹介
           </h2>
16           <p class="fs-4 text-info-emphasis mt-4">ペット優先主義が当社のこだわりです</p>
17           <div class="      "      "        mt-5">
18             <div class="      ">
19               <h3 class="bg-body-tertiary fs-4 p-2 border">優良ブリーダーとの取引</h3>
20               <p class="fs-5 text-info-emphasis          mt-4">“ペットを … </p>
21               <p class="p-2 mt-3 text-primary bg-primary-subtle mx-auto w-50              ">
                   <a class="d-block" href="#">詳細はこちら</a>
                 </p>
22             </div>
23             <div class="      ">
24               <h3 class="bg-body-tertiary fs-4 p-2 border">ペットの健康管理</h3>
25               <p class="fs-5 text-info-emphasis          mt-4">ブリーダーの … </p>
26               <p class="p-2 mt-3 text-primary bg-primary-subtle mx-auto w-50              ">
                   <a class="d-block" href="#">詳細はこちら</a>
                 </p>
27             </div>
28             <div class="      ">
29               <h3 class="bg-body-tertiary fs-4 p-2 border">店舗での取り組み</h3>
30               <p class="fs-5 text-info-emphasis          mt-4">販売すれば良い … </p>
31               <p class="p-2 mt-3 text-primary bg-primary-subtle mx-auto w-50              ">
                   <a class="d-block" href="#">詳細はこちら</a>
                 </p>
32             </div>
33           </div>
34         </div>
35       </article>
36     </main>
37   </div>
38   <script src="                    "></script>
39 </body>
40 </html>
```

問題06（難易度 ☆☆☆）

・「ペットショップPure」を紹介するWebページを作成する。
・「lib_exercise_06.html」ファイルを開き、HTMLプログラムを入力する
・プログラム作成後に「lib_exercise_06.html」ファイルを実行し、Webブラウザに表示される実行結果を確認する

［要件］

・WebページにBootstrap 5.3.0を適用し、Webページのデザイン性を高める
・HTMLタグ要素を追加し、Webページのレイアウトを整える

［設定内容］

・見出し「ペット用品」の外側の左右余白に自動を設定する
・見出し「ペット用品」の幅に親要素の50%を設定する
・２種類のペット用品の領域について、要素の並びにフレックスを設定する
・２種類のペット用品の領域について、幅に親要素の50%を設定する
・２種類のペット用品の領域について、外側の左右余白に自動を設定する
・「FOOD」の領域の幅に親要素の50%を設定する
・FOOD画像の幅に親要素の100%を設定する
・「TOYS」の領域の幅に親要素の50%を設定する
・TOYS画像の幅に親要素の100%を設定する

［実行結果］

lib_exercise_06.html

```
<!DOCTYPE html>
<html lang="ja">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link href="css/bootstrap.min.css" rel="stylesheet" type="text/css">
  <link href="css/style_pure.css" rel="stylesheet" type="text/css">
  <title>ペットショップPure（ピュア）</title>
</head>
<body>
  <div class="mx-auto w-75">
    <main>
   :
      <article>
        <div class="text-center border-top mb-5" id="item" style="padding-top: 150px;">
          <h2 class="text-primary-emphasis fw-normal [          ] p-2 border-top border-bottom
                     border-primary-subtle border-2 [    ]">
            ペット用品
          </h2>
          <p class="fs-4 text-info-emphasis mt-4 text-center">こだわりの … </p>
          [                                    ]
            <div class="mt-4 px-4 [    ]">
              <h3 class="bg-body-tertiary fs-4 p-3 mb-0 border border-dark">FOOD</h3>
              <p class="border border-dark border-top-0">
                <img src="imgs/pet_food.png" alt="" class="[      ]">
              </p>
              <p class="text-primary bg-primary-subtle p-3">
                <a class="d-block" href="#">詳細はこちら</a>
              </p>
            </div>
            <div class="mt-4 px-4 [    ]">
              <h3 class="bg-body-tertiary fs-4 p-3 mb-0 border border-dark">TOYS</h3>
              <p class="border border-dark border-top-0">
                <img src="imgs/pet_toys.png" alt=""  class="[      ]">
              </p>
              <p class="text-primary bg-primary-subtle p-3">
                <a class="d-block" href="#">詳細はこちら</a>
              </p>
            </div>
          [      ]
        </div>
      </article>
    </main>
  </div>
  <script src="js/bootstrap.min.js"></script>
</body>
</html>
```

問題07（難易度 ☆☆☆）

・「ペットショップPure」を紹介するWebページを作成する。
・「lib_exercise_07.html」ファイルを開き、HTMLプログラムを入力する
・プログラム作成後に「lib_exercise_07.html」ファイルを実行し、Webブラウザに表示される実行結果を確認する

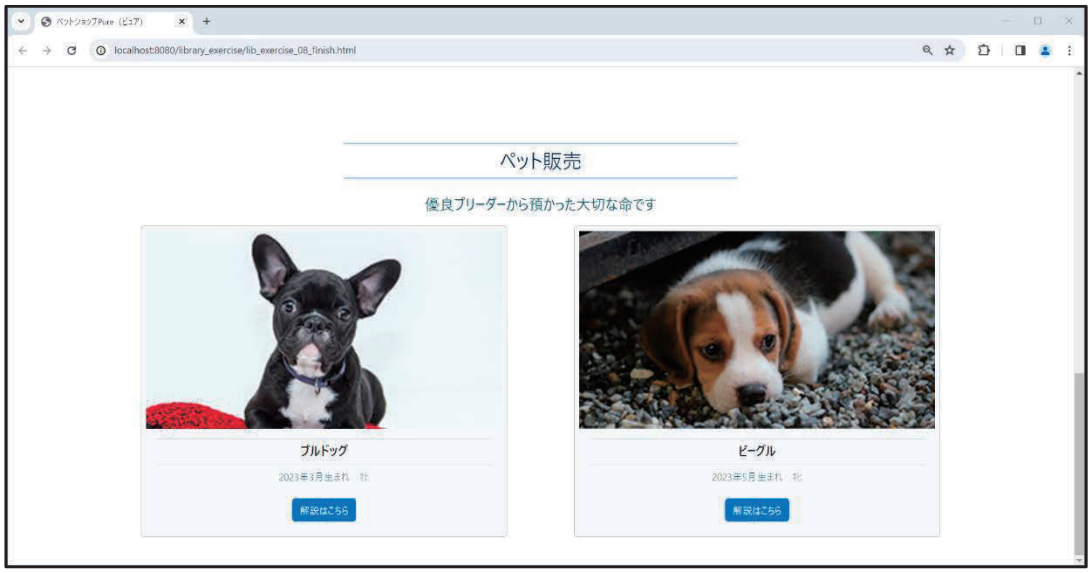
［要件］

・WebページにBootstrap 5.3.0を適用し、Webページのデザイン性を高める
・HTMLタグ要素を追加し、Webページのレイアウトを整える

［設定内容］

・２種類のペットの領域に行を設定する
・「ブルドッグ」の領域に列を設定する
・「ブルドッグ」の領域のレイアウトにカードを設定する
・ブルドッグ画像の領域に「カードの上部に画像を配置」を設定する
・「ブルドッグ」の見出しと段落の領域に「カードのボディ部」を設定する
・見出し「ブルドッグ」に「カードのタイトル」を設定する
・「ブルドッグ」の説明文の段落について、文字列に「カードのテキスト」を設定する
・「ビーグル」の領域の左側に１列分のオフセットを挿入する
・「ビーグル」の領域に列を設定する
・「ビーグル」の領域のレイアウトにカードを設定する
・ビーグル画像の領域に「カードの上部に画像を配置」を設定する
・「ビーグル」の見出しと段落の領域に「カードのボディ部」を設定する
・見出し「ビーグル」に「カードのタイトル」を設定する
・「ビーグル」の説明文の段落について、文字列に「カードのテキスト」を設定する

［実行結果］



lib_exercise_07.html

```
01 <!DOCTYPE html>
02 <html lang="ja">
03 <head>
04   <meta charset="utf-8">
05   <meta name="viewport" content="width=device-width, initial-scale=1">
06   <link href="css/bootstrap.min.css" rel="stylesheet" type="text/css">
07   <link href="css/style_pure.css" rel="stylesheet" type="text/css">
08   <title>ペットショップPure（ピュア）</title>
09 </head>
10 <body>
11   <div class="mx-auto w-75">
12     <main>
     :
54       <article>
55         <div class="container text-center border-top mb-5" id="breeder"
               style="padding-top: 150px;">
56           <h2 class="text-primary-emphasis fw-normal mx-auto p-2 border-top border-bottom
                      border-primary-subtle border-2 w-50">
             ペット販売
           </h2>
57           <p class="fs-4 text-info-emphasis mt-4 mb-4 text-center">優良ブリーダーから … </p>
58           
59             <div class="                bg-body-tertiary p-2">
60               <div class="                "><img src="imgs/dogs-bulldog.png" class="w-100"></div>
61               <div class="                ">
62                 <h3 class="                fs-5 p-2 border-top border-bottom">ブルドッグ</h3>
63                 <p class="                text-info-emphasis fw-light pb-2">2023年3月生まれ　牡</p>
64                 <button type="button" class="btn btn-primary">解説はこちら</button>
65               </div>
66             </div>
67             <div class="                bg-body-tertiary p-2">
68               <div class="                "><img src="imgs/dogs-beegle.png" class="w-100"></div>
69                 <div class="                ">
70                 <h3 class="                fs-5 p-2 border-top border-bottom">ビーグル</h3>
71                 <p class="                text-info-emphasis fw-light pb-2">2023年5月生まれ　牝</p>
72                 <button type="button" class="btn btn-primary">解説はこちら</button>
73               </div>
74             </div>
75           
76         </div>
77       </article>
78     </main>
79   </div>
80   <script src="js/bootstrap.min.js"></script>
81 </body>
82 </html>
```

問題08（難易度 ☆☆☆☆）

・「ペットショップPure」を紹介するWebページを作成する。
・「lib_exercise_08.html」ファイルを開き、HTMLプログラムを入力する
・プログラム作成後に「lib_exercise_08.html」ファイルを実行し、Webブラウザに表示される実行結果を確認する

［要件］

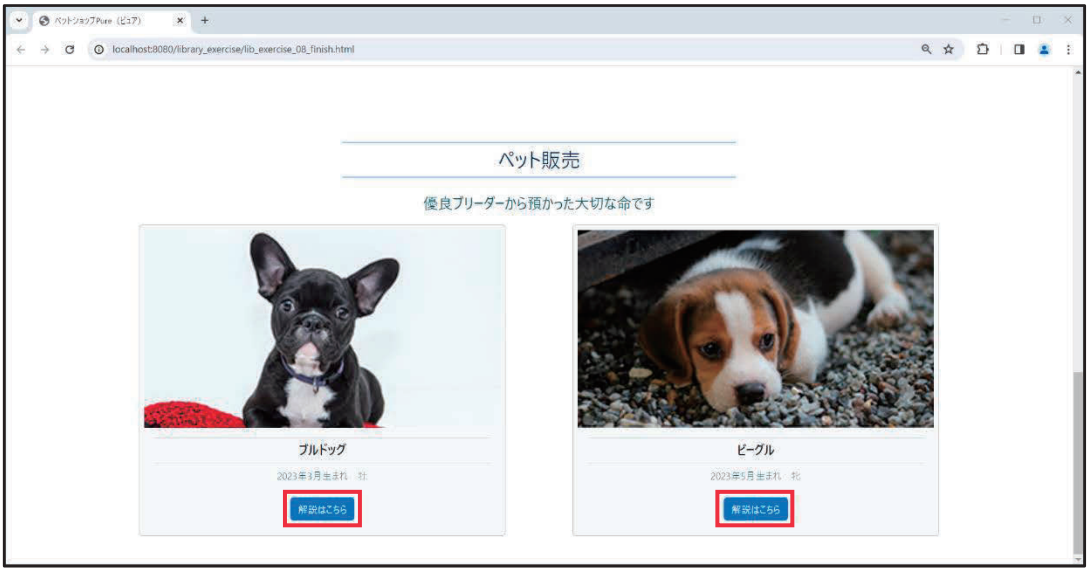
・WebページにBootstrap 5.3.0を適用し、Webページのデザイン性を高める
・HTMLタグ要素を追加し、Webページのレイアウトを整える
・Webページ内の「解説はこちら」ボタンをクリックすると、そのペットの解説がモーダル表示される

［設定内容］

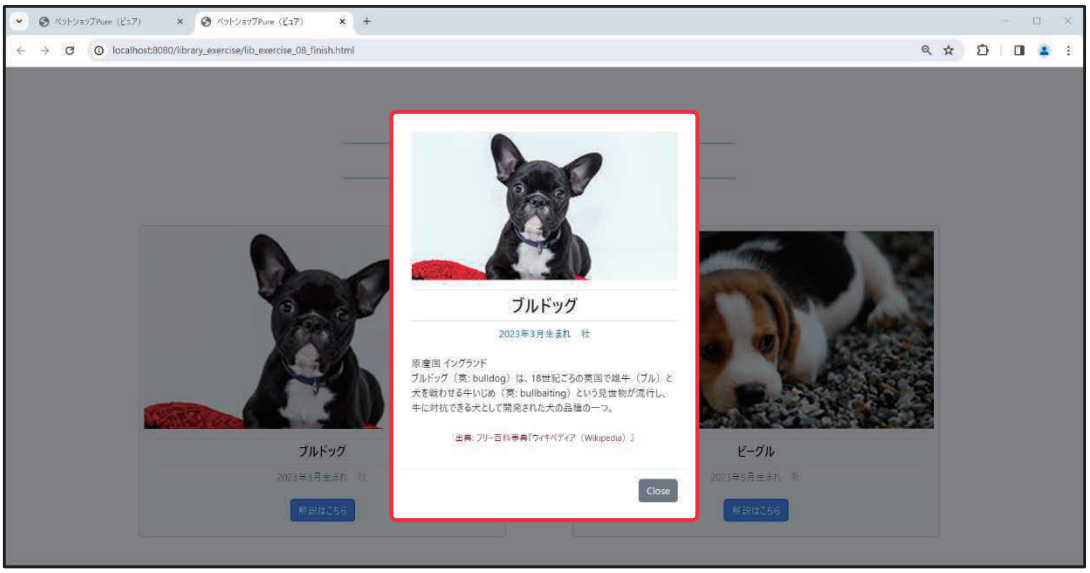
・「ブルドッグ」の「解説はこちら」ボタンにdata-bs-target属性を追加して、「ブルドッグの解説」のモーダルを設定する
・「ビーグル」の「解説はこちら」ボタンにdata-bs-target属性を追加して、「ビーグルの解説」のモーダルを設定する
・「ブルドッグの解説」のモーダルに「modal」クラスを適用する
・「ブルドッグの解説」のモーダルについて、表示方法にフェードを設定する
・「ブルドッグの解説」のモーダルにモーダルダイアログを設定する
・「ブルドッグの解説」のモーダルについて、表示位置に上下中央揃えを設定する
・「ブルドッグの解説」のモーダルについて、コンテンツ部分に「自動的にスクロールバーを表示」を設定する
・「ブルドッグの解説」のモーダルのボディ部とフッタ部の領域にモーダルコンテンツを設定する
・「ブルドッグの解説」の見出しと解説文の領域に「モーダルのボディ部」を設定する
・「ブルドッグの解説」の「Close」ボタンの領域に「モーダルのフッタ部」を設定する
・「ビーグルの解説」のモーダルに「modal」クラスを適用する
・「ビーグルの解説」のモーダルについて、表示方法にフェードを設定する
・「ビーグルの解説」のモーダルにモーダルダイアログを設定する
・「ビーグルの解説」のモーダルについて、表示位置に上下中央揃えを設定する
・「ビーグルの解説」のモーダルについて、コンテンツ部分に「自動的にスクロールバーを表示」を設定する
・「ビーグルの解説」のモーダルのボディ部とフッタ部の領域にモーダルコンテンツに設定する
・「ビーグルの解説」の見出しと解説文の領域を「モーダルのボディ部」に設定する
・「ビーグルの解説」の「Close」ボタンの領域を「モーダルのフッタ部」に設定する

［実行結果］

（通常表示）



（「解説はこちら」ボタンをクリックしてモーダル表示）



lib_exercise_08.html

```
<!DOCTYPE html>
<html lang="ja">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link href="css/bootstrap.min.css" rel="stylesheet" type="text/css">
  <link href="css/style_pure.css" rel="stylesheet" type="text/css">
  <title>ペットショップPure（ピュア）</title>
</head>
<body>
  <div class="mx-auto w-75">
    <main>
    :
                <button type="button" class="btn btn-primary" data-bs-toggle="modal"
                        data-bs-target="#            ">
                  解説はこちら
                </button>
    :
                <button type="button" class="btn btn-primary" data-bs-toggle="modal"
                        data-bs-target="#            ">
                  解説はこちら
                </button>
    :
      <div class="       " id="modalLabel1" tabindex="-1" aria-labelledby="modalLabel1"
           aria-hidden="true" data-bs-backdrop="static" data-bs-keyboard="false">
        <div class="           " text-center                    ">
          <div class="           p-3">
            <div class="mb-3"><img src="imgs/dogs-bulldog.png" class="w-100"></div>
            <h3 class="fs-3 border-top border-bottom py-2">ブルドッグ</h3>
            <p class="text-info-emphasis mb-4">2023年3月生まれ　牡</p>
            <p class="text-start">原産国　イングランド<br>ブルドッグ（英: bulldog）… </p>
            <p class="text-danger-emphasis p-2"><small>出典: … </small></p>
            <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">
              Close
            </button>
          </div>
        </div>
      </div>
      <div class="       " id="modalLabel2" tabindex="-1" aria-labelledby="modalLabel2"
           aria-hidden="true" data-bs-backdrop="static" data-bs-keyboard="false">
        <div class="           " text-center                    ">
          <div class="           p-3">
    :
```

lib_exercise_08.html（続き）

```
      :
 98                <div class="mb-3"><img src="imgs/dogs-beegle.png" class="w-100"></div>
 99                <h3 class=" fs-3 border-top border-bottom py-2">ビーグル</h3>
100                <p class="text-info-emphasis pb-2 mb-2">2023年5月生まれ　牝</p>
101                <p class="text-start">原産国　イングランド<br>ビーグル（英: Beagle）… </p>
102                <p class="text-danger-emphasis p-2 "><small>出典 … </small></p>
103              
104              
105                <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">
                     Close
                   </button>
106              
107            </div>
108          </div>
109        </div>
110      </main>
111    </div>
112    <script src="js/bootstrap.min.js"></script>
113  </body>
114  </html>
```

問題09（難易度 ☆☆☆☆）

・「ペットショップPure」を紹介するWebページを作成する。
・「lib_exercise_09.html」ファイルを開き、HTMLプログラムを入力する
・プログラム作成後に「lib_exercise_09.html」ファイルを実行し、Webブラウザに表示される実行結果を確認する

［要件］

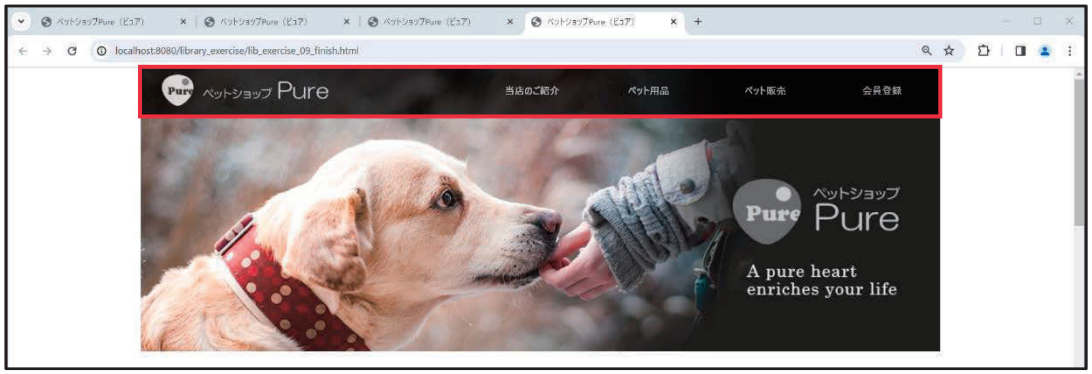
・WebページにBootstrap 5.3.0を適用し、Webページのデザイン性を高める
・HTMLタグ要素を追加し、Webページのレイアウトを整える

［設定内容］

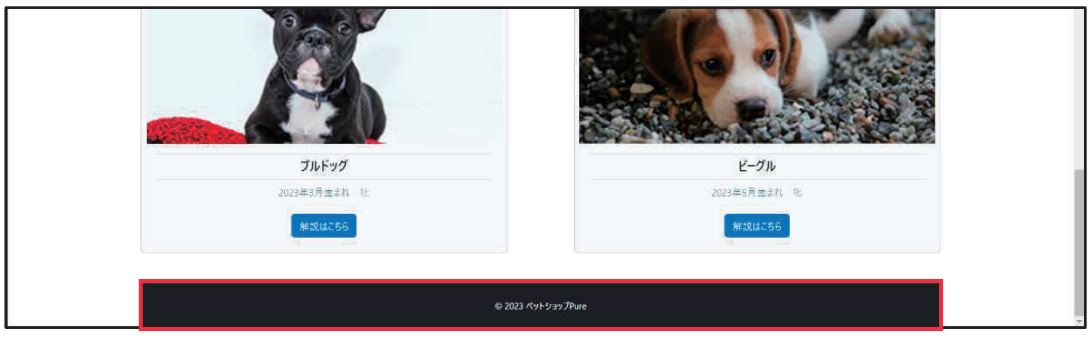
・Webページのヘッダ部分を記述する
・ヘッダ領域のレイアウトに「container」クラスを適用する
・ヘッダ領域をWebページの上部に固定する
・ヘッダ領域に不透過度75%を設定する
・ヘッダ領域の幅に親要素の75%を設定する
・ナビゲーションバーの領域に行を設定する
・ナビゲーションバーの見出しに５列分を設定する
・ナビゲーションボタンの領域に７列分を設定する
・ナビゲーションボタンの領域の要素に左右中央揃えを設定する
・ナビゲーションボタンの順序なしリストに行を設定する
・「当店のご紹介」項目、「ペット用品」項目、「ペット販売項目」項目、「会員登録」項目に３列分を設定する
・バナー画像の幅に親要素の100%を設定する
・Webページのフッタ部分を記述する
・フッタ内の領域の文字列に左右中央揃えを設定する

［実行結果］

（Webページのヘッダ部分）



（Webページのフッタ部分）



lib_exercise_09.html

```
<!DOCTYPE html>
<html lang="ja">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link href="css/bootstrap.min.css" rel="stylesheet" type="text/css">
  <link href="css/style_pure.css" rel="stylesheet" type="text/css">
  <title>ペットショップPure（ピュア）</title>
</head>
<body>
  <div class="mx-auto w-75">
    <div class="bg-black">
      <h1><img src="imgs/logo.png"></h1>
      <nav>
        <ul class="text-light pt-4" style="list-style-type: none; padding-left: 0;">
          <li><a href="#info">当店のご紹介</a></li>
          <li><a href="#item">ペット用品</a></li>
          <li><a href="#breeder">ペット販売</a></li>
          <li><a href="lib_exercise_10.html">会員登録</a></li>
        </ul>
      </nav>
    </div>
    <p><img src="imgs/top_img.png" alt="" class="w-100"></p>
    <main>
  :
    </main>
    <div class="bg-dark mt-4 p-4">
      <small class="text-light">&copy; 2023 ペットショップPure</small>
    </div>
  </div>
  <script src="js/bootstrap.min.js"></script>
</body>
</html>
```

問題10（難易度 ☆☆☆☆☆）

・「ペットショップPure」の会員登録用Webページを作成する。
・「lib_exercise_10.html」ファイルを開き、HTMLプログラムを入力する
・プログラム作成後に「lib_exercise_10.html」ファイルを実行し、Webブラウザに表示される実行結果を確認する

［要件］

・WebページにBootstrap 5.3.0を適用し、Webページのデザイン性を高める
・「氏名」、「電話番号」、「住所」、「メールアドレス」の入力値の書式を正規表現で確認し、カスタムフィールドバックスタイルで結果を表示する

［設定内容］

・会員登録フォームにclass属性を追加し、必要なBootstrapクラス名を記述する
・会員登録フォームのclass属性に、必要なBootstrapクラス名を記述する
・HTMLタグ要素は追加／消去しない
・既に記述されているBootstrapクラス名は消去しない

［実行結果］

（「氏名」、「電話番号」、「住所」、「メールアドレス」の入力値が正規表現に一致しなかった場合）

※「氏名」、「電話番号」、「住所」、「メールアドレス」テキストボックスに何も入力せずに「送信する」ボタンをクリックする
※メッセージ「〇〇は必須項目です。」は、入力値が正規表現に一致しなかった場合のみ表示される
※入力内容が送信できた場合は、未入力状態の会員登録フォームが再表示される

lib_exercise_10.html

```
01  <!DOCTYPE html>
02  <html lang="ja">
03  <head>
04    <meta charset="utf-8">
05    <meta name="viewport" content="width=device-width, initial-scale=1">
06    <link href="css/bootstrap.min.css" rel="stylesheet" type="text/css">
07    <title>ペットショップPure（ピュア）</title>
08  </head>
      :
```

lib_exercise_10.html（続き）

```
    :
09 <body>
10   <div class="p-5 mx-auto w-75 bg-warning-subtle">
11     <h1 class="fs-3 mb-4 pb-3 border-bottom">ペットショップPure（ピュア）会員登録</h1>
12     <form action="" method="post" novalidate>
13       <div class="row mb-3">
14         <label class="col-2" for="name">氏名</label>
15         <div class="col-10">
16           <input type="text" id="name" name="name"
                    placeholder="氏名を入力" pattern="…" required>
17           <div>氏名は必須項目です。</div>
18         </div>
19       </div>
20       <div class="row mb-3">
21         <label class="col-2" for="tel">電話番号</label>
22         <div class="col-10">
23           <input type="tel" id="tel" name="tel"
                    placeholder="電話番号をハイフンなしで入力" pattern="…" required>
24           <div>電話番号は必須項目です。</div>
25         </div>
26       </div>
27       <div class="row mb-3">
28         <label class="col-2" for="address">住所</label>
29         <div class="col-10">
30           <input type="text" id="address" name="address"
                    placeholder="住所を入力" pattern="…" required>
31           <div>住所は必須項目です。</div>
32         </div>
33       </div>
34       <div class="row mb-3">
35         <label class="col-2" for="email">メールアドレス</label>
36         <div class="col-10">
37           <input type="email" id="email" name="email"
                    placeholder="メールアドレスを入力" pattern="…" required>
38           <div>メールアドレスは必須項目です。</div>
39         </div>
40       </div>
41       <hr>
42       <input type="submit" value="送信する" class="btn btn-primary rounded-pill"
                 style="width: 20%;">
43       <input type="reset" value="リセット" class="btn btn-secondary rounded-pill"
                 style="width: 20%;">
44     </form>
45   </div>
46   <script src="js/bootstrap.min.js"></script>
47   <script src="js/validation.js"></script>
48 </body>
49 </html>
```

第２章　プログラミング演習　解答例

問題１

exercise01.html

```
   :
02  <html lang="ja">
03  <head>
04  <meta charset="utf-8">
   :
06</head>
   :
08  <h1>桃太郎（ももたろう）</h1>
09  <p>日本のおとぎ話の一つ。桃の実から生まれた男子「桃太郎」が、お爺さんお婆さんから<br>
10  黍団子（きびだんご）を　もらって、イヌ、サル、キジを家来にし、鬼ヶ島まで鬼を退治しに行く
    物語。</p>
   :
12  <p>以下のような粗筋のものが「標準型」となっている。<br>
13  桃から生まれた桃太郎は、老婆老爺に養われ、鬼ヶ島へ鬼退治に出征、道中遭遇する<br>
14  イヌ、サル、キジにきび団子を褒美に家来とし、鬼の財宝を持ち帰り、郷里に凱旋する</p>
15  <h3>異本</h3>
   :
19  <h3>口承文学の異伝</h3>
   :
22  <h2>その２　由来</h2>
   :
```

問題２

exercise02.html

```
   :
02  <html lang="ja">
   :
08  <h1><ruby>二<rp>(</rp><rt>に</rt><rp>)</rp>酸<rp>(</rp><rt>さん</rt>
    <rp>)</rp>化<rp>(</rp><rt>か</rt><rp>)</rp>
09  炭<rp>(</rp><rt>たん</rt><rp>)</rp>素<rp>(</rp><rt>そ</rt>
    <rp>)</rp></ruby></h1>
10  <p>炭素の酸化物の一つで、化学式がCO<sub>2</sub>と表される無機化合物
    <sup>※1</sup>である。化学式から「シーオーツー」とも呼ばれる。<br>
11  地球温暖化対策の文脈などで「カーボンフリー」「カーボンニュートラル<sup>※2</sup>」など「カ
    ーボン」が使われることがあるが、<br>
   :
15  常温常圧では無色無臭の気体<sup>※3</sup>。常圧では液体<sup>※4</sup>にならず、-79° C で
    凝華<sup>※5</sup>して固体（ドライアイス）となる。<br>
   :
17  水溶液および固体は二酸化炭素を吸収して、炭酸塩<sup>※6</sup>または炭酸水素塩を生ずる。<br>
18  高圧で二酸化炭素の飽和水溶液を冷却すると八水和物<sup>※7</sup> CO<sub>2</sub>
· 8 H<sub>2</sub> O を生ずる。</p>
```

問題３

exercise03.html

```
   :
10          <li>home</li>
11          <li>bread</li>
12          <li>Deli bread</li>
13          <li>access</li>
14          <li>contact</li>
   :
18  <img src="exercise_img/be-konmayo.jpg" width="300" height="200" alt="ベーコンマヨ">
19  <figcaption>ベーコンマヨ</figcaption>
   :
23  <img src="exercise_img/hanamaru.jpg" width="300" height="200" alt="はな丸パン">
24  <figcaption>はな丸パン</figcaption>
   :
28  <img src="exercise_img/kurowassan.jpg" width="300" height="200" alt="クロワッサン">
29  <figcaption>クロワッサン</figcaption>
   :
```

問題4

exercise04.html

```
   :
10  <ol>
11  <li>赤ずきんと呼ばれる女の子がいた。彼女はお使いを頼まれて森の向こうのおばあさんの<br>
12  家へと向かうが、その途中で一匹の狼に遭い、唆されて道草をする。</li>
13  <li>狼は先回りをしておばあさんの家へ行き、家にいたおばあさんを食べてしまう。<br>
14  そしておばあさんの姿に変装して、赤ずきんが来るのを待つ。</li>
15  <li>赤ずきんがおばあさんの家に到着すると、おばあさんに化けていた狼に赤ずきんは食べられてし
    まう。</li>
16  <li>満腹になった狼が寝入っていたところを通りかかった猟師が気付き、狼の腹の中から二人を助け
    出す。</li>
17  <li>赤ずきんは「言いつけを守らなかったから酷い目に遭った」と反省し、２度と道草をしたり知ら
    ない人の<br>
18  誘いに乗らないことを誓う。</li>
19  </ol>
   :
25  <ul>
26  <li>赤ずきんからの派生作品（英語版）</li>
27  <li>赤ずきん（曖昧さ回避）</li>
28  <li>アカンずきん</li>
29  <li>マイメロディの赤ずきん</li>
30  <li>リトル・レッド レシピ泥棒は誰だ!?</li>
31  <li>おかしな赤頭巾</li>
32  </ul>
   :
```

問題5

exercise05.html

```
   :
08      <article>
09          <h1>こぶとりじいさん</h1>
   :
13              <section>
14              <h2>異本例</h2>
15                  <section>
16                      <h3>隣の翁に瘤がないパターン</h3>
   :
23                  </section>
24                  <section>
25                      <h3>隣の翁が瘤を取ってもらうことに成功する（出羽で採取）</h3>
   :
30                  </section>
31                  </section>
   :
38    </article>
   :
```

問題6

exercise06.html

```
   :
08      <header>
   :
11              <ul>
12                  <li>ショートケーキ</li>
13                  <li>チョコレイトケーキ</li>
14              </ul>
15      </header>
16      <main>
17          <article>
18              <h2>チーズケーキの種類</h2>
   :
35          </article>
36      </main>
37  <footer>
   :
39  </footer>
   :
```

問題7

exercise07.html

```
01  <!DOCTYPE html>
02  <html lang="ja">
03    <head>
04      <meta charset="UTF-8">
05      <title>Plain bread</title>
06    </head>
07    <body>
08      <header>
09        <h1>Plain bread</h1>
10        <p>国産小麦粉を使ったオーガニックなおいしいパンを食べられる</p>
11        <nav>
12          <ul>
13            <li><a href="./">home</a></li>
14            <li><a href="item/">item</a></li>
15            <li><a href="contact/">contact</a></li>
16          </ul>
17        </nav>
18    </header>
19    <main>
20    <article>
21      <h2>焼き立てパン</h2>
22      <p>完全無農薬素材を使用した食パンを製造しています。添加物や甘味料を一切使用せずに
              自然の食材のおいしさをお届けいたします。</p>
23      <figure>
24        <img src="exercise_img/haigasyokupan.jpg" width="300" height="200" alt="胚芽ブレット">
25        <figcaption>胚芽ブレット</figcaption>
26      </figure>
27    </article>
28    </main>
29    <footer>
30      <p>Copyright &copy; 2015 Plain bread Corporation. All Rights Reserved.</p>
31    </footer>
32  </body>
33  </html>
```

itemフォルダ・index.html

```
  :
11      <nav>
12        <ul>
13          <li><a href="../exercise07.html">home</a></li>
14          <li><a href="./">item</a></li>
15          <li><a href="../contact/">contact</a></li>
16        </ul>
17      </nav>
  :
```

contactフォルダ・index.html

```
  :
11      <nav>
12        <ul>
13          <li><a href="../exercise07.html">home</a></li>
14          <li><a href="../item/">item</a></li>
15          <li><a href="./">contact</a></li>
16        </ul>
17      </nav>
  :
```

問題8

exercise08.html

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="UTF-8">
    <title>ケーキ屋 デリシャス</title>
  </head>
  <body>
    <header>
      <h1><img src="exercise_img/exercise-07.gif" width="350" height="76" alt="delicious"></h1>
      <p>自然素材をを使った、おいしいケーキ、国産小麦粉・純生クリーム、オーガニックチョコレートなど安心して食べられる素材を使用</p>
      <nav>
        <ul>
          <li>home</li>
          <li>concept</li>
          <li>menu</li>
          <li>news</li>
        </ul>
      </nav>
    </header>
  <main>
  <article>
    <h2>Menu</h2>
    <section>
      <h3>ショートケーキ</h3>
      <p>国産の卵を使用したこだわりのスポンジとなめらかなクリームに極上のイチゴを乗せました。<br>
      パティシエ自慢のショートケーキです。</p>
      <figure>
        <img src="exercise_img/syouto.jpg" width="200" height="200" alt="イチゴのショートケーキ">
        <figcaption>イチゴのショートケーキ</figcaption>
      </figure>
    </section>
    <section>
      <h3>アップルタルト</h3>
        <p>国産小麦を使用したこだわりタルトに完熟のリンゴを使用し<br>
        甘酸っぱく仕上げました。サクサクの生地をお楽しみください。</p>
        <figure>
          <img src="exercise_img/appurutaruto.jpg" width="200" height="200" alt="アップルタルト">
          <figcaption>アップルタルト</figcaption>
        </figure>
      </section>
      <section>
        <h3>イチゴのムース</h3>
```

```
43          <p>完熟したイチゴをつぶした手作りのジャムを使用したムースです。<br>
44          イチゴの風味たっぷりな、なめらかな口当たりのムースとなっております。</p>
45        <figure>
46          <img src="exercise_img/itigonomu-su.jpg" width="200" height="200" alt="イチゴのムース">
47          <figcaption>イチゴのムース</figcaption>
48        </figure>
49      </section>
50    </article>
51    </main>
52    <footer>
53      <p>Copyright &copy; 2015 delicious Corporation. All Rights Reserved.</p>
54    </footer>
55    </body>
56  </html>
```

問題9

exercise09.html

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="UTF-8">
    <title>ごうか美術館</title>
  </head>
<body>
    <header>
      <h1>ごうか美術館</h1>
      <p>この世の中で一番美しい物を収集した展示会「GOUKA」</p>
      <nav>
        <ul>
          <li>home</li>
          <li>exhibition</li>
          <li>access</li>
        </ul>
      </nav>
    </header>
  <main>
  <article>
    <h2>入場料</h2>
      <table border="1">
        <thead>
          <tr>
            <th>入館料</th>
            <th>前売券</th>
            <th>当日券</th>
          </tr>
        </thead>
        <tbody>
          <tr>
            <td>一般</td>
            <td>&yen;3,200-</td>
            <td>&yen;3,500-</td>
          </tr>
          <tr>
            <td>大学生</td>
            <td>&yen;2,200-</td>
            <td>&yen;2,500-</td>
          </tr>
          <tr>
            <td>小中高生</td>
            <td>&yen;500-</td>
            <td>&yen;600-</td>
          </tr>
        </tbody>
        <tfoot>
```

HTML/CSS

```
48            <tr>
49              <td>備考</td>
50              <td colspan="2">割引券の重複適用はできません。</td>
51            </tr>
52          </tfoot>
53        </table>
54      </article>
55    </main>
56    <footer>
57      <p>Copyright &copy; 2023 gouka Corporation. All Rights Reserved.</p>
58    </footer>
59  </body>
60  </html>
```

問題10

Exercise10.html

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="UTF-8">
    <title>ごおか文具店</title>
  </head>
<body>
  <header>
    <h1>ごおか文具店</h1>
    <nav>
      <ul>
        <li>home</li>
        <li>item</li>
        <li>access</li>
      </ul>
    </nav>
  </header>
  <main>
  <article>
    <h2>応募フォーム</h2>
      <form action="Apply.php" method="post" autocomplete="off">
        <table border="1">
          <tbody>
            <tr>
              <td><label for="id_name">名前　　 &colon; </label></td>
              <td><input type="text" name="name" placeholder="名前を入力してください"
maxlength="10" id="id_name"></td>
            </tr>
            <tr>
              <td><label for="mail">メールアドレス &colon;  </label></td>
              <td><input type="email" name="mail" placeholder="メールアドレスを入力してく
ださい" required id="mail"></td>
            </tr>
            <tr>
              <td>性別　　 &colon;  </td>
              <td><label><input type="radio" name="seibetu" value="男性"> 男性
</label>  <label><input type="radio" name="seibetu" value="女性">女性</label></td>
            </tr>
            <tr>
              <td>年代　　 &colon;  </td>
              <td><label><input type="radio" name="age" value="20代"> 20代
</label>  <label><input type="radio" name="age" value="30代"> 30代</label> 
<label><input type="radio" name="age" value="40代"> 40代</label>  <label><input
type="radio" name="age" value="50代以上"> 50代以上</label></td>
            </tr>
            <tr>
              <td><label for="selects">懸賞コース　 &colon;  </td>
```

HTML/CSS

```
42                <td><select name="course" id="selects">
43                      <option value="Aコース">Aコース</option>
44                      <option value="Bコース">Bコース</option>
45                      <option value="Cコース">Cコース</option>
46                </select></td>
47              </tr>
48              <tr>
49                <td><label for="textarea01">お問合せ内容</label></td>
50                <td><textarea name="otoiawase" rows="10" cols="80"
  id="textarea01"></textarea></td>
51              </tr>
52            </tbody>
53            <tfoot>
54              <tr>
55                <td colspan="2"><button type="submit" name="submit">送信</button>  <button
  type="reset" name="reset">リセット</button></td>
56              </tr>
57            </tfoot>
58          </table>
59        </form>
60    </article>
61    </main>
62    <footer>
63      <p>Copyright &copy; 2023 gouka Corporation. All Rights Reserved.</p>
64    </footer>
65  </body>
66  </html>
```

第４章　プログラミング演習　解答例

問題１

exercise01.html

```
  :
06<style>
  :
08        color: #808080;
09        background-color: #ebf6f7;
  :
11    h1{
12        font-size: 28px;
  :
14        text-shadow: 1px 2px 2px #808080;
  :
17        font-size: 18px;
18        font-weight: normal;
  :
21    p{
22        font-size: 14px;
23        background-color: #fdeff2;
  :
25</style>
  :
29      <h2 style="color: #165e83;">色名としての青</h2>
30      <p style="text-align: left;">青という基本色名は、その他多くの固有色名を総称
  :
34      <h2 style="color: #ba2636 ;">基本色名としての赤</h2>
35      <p style="text-align: center;">丹（タン）が色を名指すときは赭土（シャド）、赤土の色の意
  :
38      <h2 style="color: #38b48b ;">緑という色名</h2>
39      <p style="text-align: right;">緑に相当する色はかなり広範に及ぶ色の総称であるが、<br>
```

問題2

exercise02.html

```
  :
08          width: 100%;
  :
15          width: 300px;
  :
17          margin-right: 50px;
18            border: solid 1px #808080;
  :
29          <div style="display: flex; flex-direction: row;">
30              <div class="sizes">
  :
36              <div class="sizes">
  :
42</div>
  :
```

問題3

index.html

```
  :
06        <link rel="stylesheet" href="css/exercise_03.css" type="text/css">
  :
```

exercise_03.css

```
  :
03          width:900px;
04          margin: 0 auto;
  :
12          background-image: url(../images/aozora.jpg);
13          background-size: cover;
  :
25          background-image: url(../images/css_ensyu-3-1.jpg),url(../images/css_ensyu-3-2.jpg);
26          background-repeat: no-repeat,no-repeat;
27          background-position: left top,right bottom;
  :
31          grid-template-columns:1fr 1fr;
  :
39          background-repeat: no-repeat;
40          background-size:contain;
  :
42    .container div:first-child{
  :
45    .container div:last-child{
  :
```

問題4

exercise_04.css

```
  :
04        border-collapse: collapse;
05        border: solid 1px #808080;
  :
13        width: 150px;
14        color: #fff;
15        background-color: #808080;
16        box-sizing: border-box;
17        padding: 10px 20px;
18        border-right: solid 1px #fff;
}
20        table thead tr th:nth-of-type(7){
  :
29        vertical-align: top;
30        font-size: 35px;
31        font-weight: bold;
32        border: solid 1px #808080;
  :
34        table tbody tr td:nth-of-type(1){
  :
37        table tbody tr td:nth-of-type(7){
  :
41        content: ”  5”;
42        display: block;
  :
48      border-radius: 100px;
  :
```

問題5

index.html

```
  :
08  <body>
09    <div class="wrapper">
10      <header>
11        <h1>駄菓子屋本舗</h1>
  :
67    </footer>
68    </div>
69  </body>
70  </html>
  :
```

exercise_05.css

```
  :
03          width: 850px;
04          margin: 0 auto;
  :
07          border: 1px solid #808080;
08          border-collapse: collapse;
  :
10          table tr:nth-of-type(2n),table tfoot tr{
  :
14          padding: 15px 20px;
15          vertical-align: middle;
  :
20          table tr:first-child input{
  :
28          input:focus,textarea:focus{
  :
31    button:focus{
32          outline: solid 2px #b7282e;
  :
```

問題6

index.html

```
  :
06    <link rel="stylesheet" href="css/style.css" type="text/css">
  :
09      <div class="wrapper">
  :
11          <div class="header_area">
  :
22          </div>
  :
117     </div>
  :
```

style.css

```
  :
33          width: 900px;
34          margin: 0 auto;
  :
38          display: flex;
39          flex-direction: row;
40          justify-content: space-between;
  :
46          font-family: "Oswald-Medium";
47          background-image: url(../images/cafe_rogo.gif);
48          background-repeat: no-repeat;
49          background-position: 10px 0px;
  :
61          font-family: "FiraSans-Medium";
  :
65          text-decoration: none;
66          display: block;
  :
69    .header_area ul li a:hover{
  :
```

問題7

index.html

```
  :
23          <div class="main_visual">
  :
27              <div class="visual_title">
  :
30              </div>
31          </div>
  :
```

style.css

```
  :
79    .main_visual{
80          position: relative;
81          box-shadow:0px 5px 3px -3px #320c00;
82    }
83    .visual_title{
84          width: 350px;
85          color: #fff;
86          background-color: rgba(50, 12, 0, .7);
87          position: absolute;
88          left: 50px;
89          bottom: 30px;
90    }
91    .visual_title h2{
92          font-size: 27px;
93          margin: 20px 50px;
94          font-family: "NotoSansJP-ExtraLight";
95    }
96    .visual_title h2 span{
97          display: block;
98          margin-left: 50px;
99    }
100    .visual_title p{
101          font-size: 15px;
102          font-family: "NotoSansJP-ExtraLight";
103          text-align: center;
104    }
105    .visual_title p span{
106          display: block;
107          border-top: dotted 2px #fff;
108          margin: 8px 0px 30px;
109          padding-top: 10px;
110    }
111    /*----------ヘッダーの設定*/
  :
```

問題8

index.html

```
  :
36                          <div class="contents_upward">
  :
38                              <div class="text_outer">
  :
42                              </div>
  :
47                          <div class="band">
  :
49                              <div class="text_outer">
  :
54                              </div>
  :
59                          </div>
  :
61                              <div class="text_outer">
  :
65                              </div>
  :
70                          </div>
  :
```

style.css

```
  :
114 /*----------コンテンツの設定*/
115   header{
116       margin-bottom: 50px;
117 }
118   main article h2{
119       font-family: "FiraSans-Medium";
120       padding-bottom: 5px;
121       border-bottom: solid 2px #320c00;
122       margin-bottom: 75px;
123 }
124   .contents_upward{
125       width: 100%;
126 }
127   .contents_upward section{
128       display: flex;
129       flex-direction: row;
130       width: 700px;
131       margin: 0 auto;
132       margin-bottom: 75px;
133 }
134   .text_outer{
135       width: 75%;
136       margin-right: 25px;
```

HTML/CSS

```
}
  .text_outer h3{
      font-size: 16px;
      margin: 0;
      border-bottom: solid 1px #320c00;
}
  .text_outer p{
      font-size: 13px;
}
  .band{
      padding: 20px;
      background-color: rgba(52, 12, 0, .1);
      margin-bottom: 75px;
}
  .band section{
      flex-direction: row-reverse;
      padding: 20px;
      margin-bottom: 0;
}
  .band section .text_outer{
      margin: 0 0 0 25px;
}
  :
```

問題9

index.html

```
  :
72                <div class="under_contents">
  :
105                </div>
  :
107                <ul class="topics">
108                    <li>2023.9.15　秋のウインタフェア開催<span>期限：９月15日～10月1日まで</span></li>
109                    <li>2023.7.10　夏メニュー限定かき氷始めました。<span>抹茶、イチゴ、ブルーハワイ、3種類</span></li>
120                    <li>2023.5.01　期間限定極上のcoffee販売<span>限定25杯分　お一人1杯限定です。</span></li>
  :
```

style.css

```
  :
161     .under_contents{
162         width: 700px;
163         margin: 0 auto 100px;
164         display: grid;
165         grid-template-columns: 1fr 1fr 1fr 1fr;
166         gap: 15px;
167 }
168     .under_contents figure figcaption{
169         text-align: center;
170 }
171 /*----------コンテンツの設定*/
172 /*----------トピックスの設定*/
173     main article h2:nth-of-type(2),
174     main article h2:nth-of-type(3){
175         margin: 0 0 50px;
176 }
177     .topics{
178         width: 500px;
179         font-size: 13px;
180         height: 75px;
181         overflow: auto;
182         margin: 0 auto;
183         background-color:#f8f8f8 ;
184 }
185     .topics li:nth-of-type(1)::before{
186         content: "NEW ";
187         color: #e83929;
188         margin-left: -35px;
189 }
190     .topics li{
191         margin-left: 50px;
```

HTML/CSS

```
192 }
193     .topics li span{
194         display: block;
195         margin-left: 79px;
196 }
197 /*----------トピックスの設定*/
  :
```

問題10

index.html

```
  :
118   <div class="footer_content">
119       <div class="footer_left">
  :
125       </div>
126       <div class="footer_right">
127           <ul class="clearfix">
  :
134           <ul class="clearfix">
  :
139       </div>
140   </div>
  :
```

style.css

```
  :
200 /*----------フッターの設定*/
201   footer{
202       background-color: #303030;
203       color: #c0ad98;
204 }
205   .footer_content{
206       display: flex;
207       flex-direction: row;
208       align-items: flex-start;
209       margin: 50px auto 10px;
210       padding-top: 25px;
211       width: 660px;
212       font-size: 14px;
213 }
214   .footer_left{
215       width: 300px;
216       margin-right: 57px;
217 }
218   .footer_left h2{
219       margin: 5px 0px;
220 }
221   .footer_left p{
222       margin: 5px 0px;
223 }
224   .footer_right{
225       width: 300px;
226       margin-top: 35px;
227       border-left: dotted 3px #836543;
228 }
229   .footer_right ul{
230       margin-left: 57px;
```

HTML/CSS

```
231 }
232   .footer_right ul:last-child{
233       margin-top: 10px;
234 }
235   .footer_right ul li{
236       padding-right: 15px;
237       float: left;
238 }
239   .footer_content+p{
240       padding: 10px;
241       margin: 0;
242       text-align: center;
243       color: #836543;
244 }
245   .clearfix::after{
246       content: "";
247       display: block;
248       clear: both;
249 }
250 /*----------フッターの設定*/
  :
```

第１章　プログラミング演習　解答例

問題１

exercise1.js

```
      :
09  for(let i = 0; i < names.length; i++) {
      :
11    totalAmount = prices[i] * stocks[i];
12    console.log("在庫総額は" + totalAmount.toLocaleString("ja-JP") + "円です。");
13    totalInventory = totalInventory + totalAmount;
      :
18  console.log("全機種の在庫総額は" + totalInventory.toLocaleString("ja-JP") + "円です。");
      :
24  for(let j = 0; j < names.length; j++) {
      :
26      console.log(names[j] + "は" + (15 - stocks[j]) + "台発注が必要です。");
27      totalAmount = prices[j] * (15 - stocks[j]);
      :
29      console.log("発注金額は" + totalAmount.toLocaleString("ja-JP") + "円です。");
      :
```

問題２

exercise2.js

```
      :
08    if(i % 400 == 0) {
09      leapYear.push(i);
10    } else if(i % 100 != 0 && i % 4 == 0) {
11      leapYear.push(i);
12    }
      :
20    if(leapYear[j] < nowYear) {
21      console.log((j + 1) + "回目のうるう年は" + leapYear[j] + "年でした。【過去】");
22    } else {
23      console.log((j + 1) + "回目のうるう年は" + leapYear[j] + "年です。");
24    }
      :
```

JavaScript

問題３

exercise3.js

```
    :
for(let i = 0; i < rCode.length; i++) {
  totalFee = totalFee + repairFee(rCode[i], rUnit[i], rPeople[i]);
}
    :
  switch(code) {
  case 1:
    price = 2000;
    break;
  case 2:
    price = 3500;
    break;
  case 3:
    price = 5800;
    break;
  case 4:
    price = 9800;
    break;
  case 5:
    price = 15000;
  }
    :
```

問題４

exercise4.js

```
'use strict';
let sensorData = "a,b,e,c,e,a,c,b,e,c,a,b,b,a,c,b,b,b,c,a";

//センサーの出力データを集計する
let aCnt = 0;           //aの出力回数
let bCnt = 0;           //bの出力回数
let cCnt = 0;           //cの出力回数
let eCnt = 0;           //e（エラーコード）の出力回数
let splitData = sensorData.split(",");
for(let data of splitData) {
  if(data == "a") {
    aCnt++;
  } else if(data == "b") {
    bCnt++;
  } else if(data == "c") {
    cCnt++;
  } else if(data == "e") {
    eCnt++;
  }
}

//エラーコードの出力回数に合わせてメッセージと出力回数を出力する
if(eCnt <= 4) {
  if(eCnt == 0) {
    console.log("センサーの出力データは優良です。");
  } else {
    console.log("センサーの出力データは許容範囲内です。");
  }
  console.log("aの出力回数：" + aCnt + "回");
  console.log("bの出力回数：" + bCnt + "回");
  console.log("cの出力回数：" + cCnt + "回");
} else {
  console.log("センサーの出力データは許容範囲外のため破棄してください。");
}
console.log("eの出力回数：" + eCnt + "回");
```

問題５

exercise5.js

```
'use strict';
const countries = [["Chiiina", "Uniiited Kingdom"],
                   ["Geeermany", "Beeelgium", "Switzeeerland"],
                   ["United Staaates", "Japaaan", "Fraaance", "Portugaaal"]];

//国名の誤り部分を正規表現による置換によって修正する
for(let i = 0; i < countries.length; i++) {
  for(let j = 0; j < countries[i].length; j++) {
    switch(i) {
    case 0:
      countries[i][j] = countries[i][j].replace(/iii/g, "i");
      break;
    case 1:
      countries[i][j] = countries[i][j].replace(/eee/g, "e");
      break;
    case 2:
      countries[i][j] = countries[i][j].replace(/aaa/g, "a");
    }
  }
}

//修正後の国名を出力する
for(let i = 0; i < countries.length; i++) {
  console.log((i + 1) + "行目");
  for(let j = 0; j < countries[i].length; j++) {
    console.log("  " + countries[i][j]);
  }
}
```

第２章　プログラミング演習　解答例

問題１

exercise1.js

```
    :
16  let sales = price[0] * unit[0];
    :
20    sumSales = sumSales + sales;
    :
22      maxSales = sales;
23      max = i;
    :
26      minSales = sales;
27      min = i;
    :
30  avgSales = Math.floor(sumSales / typeNo.length);
    :
35  console.log("売上最高額の型番は" + typeNo[max] + "で、売上額は"
36              + maxSales.toLocaleString("ja-JP") + "円です。");
37  console.log("売上最低額の型番は" + typeNo[min] + "で、売上額は"
38              + minSales.toLocaleString("ja-JP") + "円です。");
```

問題２

exercise2.js

```
    :
10    while(examNo[i] != passNo[mi] && le < ri) {
11      if(examNo[i] < passNo[mi]) {
12        ri = mi - 1;
13      } else {
14        le = mi + 1;
15      }
16      mi = Math.floor((le + ri) / 2);
17    }
    :
20    if(examNo[i] == passNo[mi]) {
21      console.log("受験番号" + examNo[i] + "は合格です。");
22    } else {
23      console.log("受験番号" + examNo[i] + "は不合格です。");
24    }
    :
```

問題３

exercise3.js

```
      :
09    let k = i;
10    for(let j = i + 1; j < names.length; j++) {
11      if(stocks[k] < stocks[j]) {
12        k = j;
13      }
14    }
      :
29    let work;
30    work = arr[x];
31    arr[x] = arr[y];
32    arr[y] = work;
      :
```

問題４

exercise4.js

```
01  'use strict';
02  const scores = [95, 43, 7, 64, 56, 99, 68, 77, 4,18, 76, 6, 73, 74, 30, 31, 83, 33, 32, 54];
03
04  //試験の得点を降順に整列する（基本交換法）
05  for(let i = 0; i < scores.length - 1; i++) {
06    for(let j = scores.length - 1; j > i; j--) {
07      if(scores[j - 1] < scores[j]) {
08        let work = scores[j - 1];
09        scores[j - 1] = scores[j];
10        scores[j] = work;
11      }
12    }
13  }
14
15  //試験の上位10件の得点について、順位と得点を出力する
16  console.log("上位10件：");
17  for(let i = 0; i < scores.length && i < 10; i++){
18    console.log("第" + (i + 1) + "位の得点は" + scores[i] + "点です。");
19  }
```

JavaScript

問題５

exercise5.js

```
'use strict';
const passNo = [438, 889, 333, 315, 599, 891, 326, 235, 395, 656];

//試験の合格者の受験番号を昇順に整列する（クイックソート）
quickSort(passNo, 0, passNo.length - 1);

//整列後の、試験の合格者の受験番号を出力する
console.log("合格者の受験番号：");
for(let pn of passNo) {
  console.log(pn);
}

//クイックソートを行うユーザ定義関数
function quickSort(arr, le, ri) {
  if(le < ri) {
    let mi = split(arr, le, ri);
    quickSort(arr, le, mi - 1);
    quickSort(arr, mi + 1, ri);
  }
}

//基準値より大きいグループと小さいグループに分けるユーザ定義関数
function split(arr, le, ri) {
  let base = arr[Math.floor((le + ri) / 2)];
  while(le < ri) {
    while(arr[le] < base) {
      le++;
    }
    while(arr[ri] > base) {
      ri--;
    }
    if(le < ri) {
      let work = arr[le];
      arr[le] = arr[ri];
      arr[ri] = work;
    }
  }
  return le;
}
```

第３章　プログラミング演習　解答例

問題１

exercise1.js

```
      :
08    let bannerImage = document.getElementsByClassName("header");
      :
10    var hour = Number(now.getHours());
      :
13    } else if(hour >= 11 && hour <= 14) {
14      bannerImage[0].style.backgroundImage = "url(" + filePath[1] + ")";
15    } else if(hour >= 15 && hour <= 18) {
16      bannerImage[0].style.backgroundImage = "url(" + filePath[2] + ")";
17    } else if(hour >= 19 && hour <= 22) {
18      bannerImage[0].style.backgroundImage = "url(" + filePath[3] + ")";
      :
20      bannerImage[0].style.backgroundImage = "url(" + filePath[4] + ")";
      :
```

問題２

exercise2.js

```
      :
15    let flowers = document.getElementsByClassName("flowers");
16    for(let elem of flowers) {
17      styleChange(elem);
18    }
      :
23    elem.style.color = "darkgoldenrod";
24    elem.style.fontWeight = "bold";
25    elem.style.textShadow = "1px 1px 2px black";
      :
```

問題３

exercise3.js

```
    :
  for(let elem of flower) {
    if(elem.checked) {
      words.push("好きな花の名前：" + elem.value);
      break;
    }
  }
  words.push("好きな色：" + color[0].options[color[0].selectedIndex].value);
  words.push("お問合せ内容：" + otoiawase[0].value);
    :
```

問題４

exercise4.js

```
'use strict';
const filePath = {あじさい : "images/hydrangea_header.jpg",
                  ひまわり : "images/sunflower_header.jpg",
                  きんもくせい : "images/osmanthus_header.jpg",
                  コスモス : "images/cosmos_header.jpg",
                  その他 : "images/header-images.jpg"};          //画像ファイルのパス

function formFunction() {
  //Webページのバナー画像を「好きな花の名前」ラジオボタンに合わせて変更する
  let flower = document.getElementsByName("flower");
  let bannerImage = document.getElementsByClassName("header");
  let flowerName = "その他";
  for(let i = 0; i < flower.length; i++) {
    if(flower[i].checked) {
      flowerName = flower[i].value;
      break;
    }
  }
  bannerImage[0].style.backgroundImage = "url(" + filePath[flowerName] + ")";

  //フォームの送信をキャンセルする
  return false;
}
```

問題５

exercise5.js

```
'use strict';

function formFunction() {
  //修正用フォームに入力された送料を取得する
  let table = document.getElementsByClassName("table-contents");
  let update = document.getElementsByClassName("update");

  //入力された送料が空欄の場合は、修正用フォームに0(円)を表示する
  for(let i = 0; i < update.length; i++) {
    if(update[i].value == "") {
      update[i].value = 0;
    }
  }

  //全国価格別送料一覧表を更新する
  changeTableData(table[0].rows[1].cells[1], update[0].value);
  changeTableData(table[0].rows[1].cells[2], update[1].value);
  changeTableData(table[0].rows[1].cells[3], update[2].value);
  changeTableData(table[0].rows[2].cells[1], update[3].value);
  changeTableData(table[0].rows[2].cells[2], update[4].value);
  changeTableData(table[0].rows[3].cells[1], update[5].value);

  //フォームの送信をキャンセルする
  return false;
}

//全国価格別送料一覧表のセルを更新するユーザ定義関数
function changeTableData(cell, value) {
  let price = Number(value);
  cell.innerHTML = price.toLocaleString("ja-JP") + "円";
  if(value == 0) {
    cell.classList.add("err");
  } else {
    cell.classList.remove("err");
  }
}
```

第４章　プログラミング演習　解答例

問題１

exercise1.js

```
      :
16    alert(flower + "の花言葉¥n" + flowers[flower]);
      :
```

index.html

```
      :
72      <img src="images/hana_01.jpg" alt="アネモネの花"
            width="110" height="110" onclick="flowerFunction('アネモネ');">
      :
76      <img src="images/hana_02.jpg" alt="チューリップの花"
            width="110" height="110" onclick="flowerFunction('チューリップ');">
      :
80      <img src="images/hana_03.jpg" alt="ニッコウキスゲの花"
            width="110" height="110" onclick="flowerFunction('ニッコウキスゲ');">
      :
84      <img src="images/hana_04.jpg" alt="テッサの花"
            width="110" height="110" onclick="flowerFunction('テッサ');">
      :
88      <img src="images/hana_05.jpg" alt="ゼラニウムの花"
            width="110" height="110" onclick="flowerFunction('ゼラニウム');">
      :
92      <img src="images/hana_06.jpg" alt="クレマチス　プリンセス・ダイアナの花"
            width="110" height="110" onclick="flowerFunction('クレマチス');">
      :
96      <img src="images/hana_07.jpg" alt="ベゴニアの花"
            width="110" height="110" onclick="flowerFunction('ベゴニア');">
      :
100     <img src="images/hana_08.jpg" alt="ムギワラギクの花"
            width="110" height="110" onclick="flowerFunction('ムギワラギク');">
      :
```

問題２

exercise2.js

```
      :
06    for(let i = 0; i < buttons.length; i++) {
07      buttons[i].classList.remove("linkErr");
08    }
      :
20    if(name != "contact") {
21      alert("申し訳ございません。このページは現在メンテナンス中です。");
22      elem.classList.add("linkErr");
23    } else {
24      document.location.href = "contact.html";
25    }
      :
```

問題３

exercise3.js

```
      :
06    for(let i = 0; i < flowers.length; i++) {
07      flowers[i].style.border = null;
08    }
      :
11    elem.style.border = "7px solid #00FF00";
      :
```

index.html

```
      :
42    <img src="images/kinmokusei170x135.jpg" alt="金木犀の花" width="170" height="135"
          class="flowerImg" onclick="flowerFunction2(this);">
      :
52    <img src="images/ajisai170x135.jpg" alt="紫陽花の花" width="170" height="135"
          class="flowerImg" onclick="flowerFunction2(this);">
      :
62    <img src="images/hanamizuki170x135.jpg" alt="花水木の花" width="170" height="135"
          class="flowerImg" onclick="flowerFunction2(this);">
      :
```

JavaScript

問題４

exercise4.js

```
'use strict';

function inputDataCheck() {
  //プログレスバーの値に 0 を代入する
  let progress = document.getElementsByName("progress");
  progress[0].value = 0;

  //氏名、メールアドレス、電話番号、お問合せ内容の文字列長が 0 より大きい場合は、
  //それぞれプログレスバーの値に 17%を加算する
  let name = document.getElementsByName("name");
  let mail = document.getElementsByName("mail");
  let telephone = document.getElementsByName("telephone");
  let otoiawase = document.getElementsByName("otoiawase");
  let elems = [name[0], mail[0], telephone[0], otoiawase[0]];
  for(let elem of elems) {
    if(elem.value.length > 0) {
      progress[0].value += 17;
    }
  }

  //好きな花の名前が選択されている場合は、プログレスバーの値に 17%を加算する
  let flowers = document.getElementsByName("flower");
  for(let i = 0; i < flowers.length; i++) {
    if(flowers[i].checked) {
      progress[0].value += 17;
    }
  }

  //好きな色が選択されている場合は、プログレスバーの値に 15%を加算する
  let colors = document.getElementsByName("color");
  if(colors[0].selectedIndex >= 1) {
    progress[0].value += 15;
  }

  //プログレスバーの値が 100%の場合は、ポップアップメッセージを表示する
  if(progress[0].value == 100) {
    alert("送信準備が整いました！");
  }
}
```

問題５

exercise5.js

```
'use strict';
const errorMsg = { 氏名エラー           :"氏名の書式に合わせて入力してください。¥n"
                                        +"(例) 聖徳 太子",
                   メールアドレスエラー :"メールアドレスの書式に合わせて入力してください。¥n"
                                        +"(例) syoutoku@hoge.co.jp",
                   電話番号エラー       :"電話番号の書式に合わせて入力してください。¥n"
                                        +"(例) 0120-999-888" };          //エラーメッセージ

function formFunction() {
  //チェックに0を代入する
  let check = 0;

  //氏名の書式に一致しない場合は、ポップアップメッセージを表示し、
  //チェックに100を加算する
  let names = document.getElementsByName("name");
  let regex = /^.+[　]+.+$/;
  if(!regex.test(names[0].value)) {
    alert(errorMsg["氏名エラー"]);
    check += 100;
  }

  //メールアドレスの書式に一致しない場合は、ポップアップメッセージを表示し、
  //チェックに10を加算する
  let mails = document.getElementsByName("mail");
  regex = /[¥w¥d_-]+@[¥w¥d_-]+¥.[¥w¥d._-]+/;
  if(!regex.test(mails[0].value)) {
    alert(errorMsg["メールアドレスエラー"]);
    check += 10;
  }

  //電話番号の書式に一致しない場合は、ポップアップメッセージを表示し、
  //チェックに1を加算する
  let telephones = document.getElementsByName("telephone");
  regex = /¥d{2,4}-¥d{3,4}-¥d{3,4}/;
  if(!regex.test(telephones[0].value)) {
    alert(errorMsg["電話番号エラー"]);
    check += 1;
  }

  //チェックが0に等しい場合はフォームを送信し、
  //そうでない場合はフォームの送信をキャンセルする
  if(check == 0) {
    return true;
  } else {
    return false;
  }
}
```

第２章　プログラミング演習　解答例

問題01

lib_exercise_01.html

```
      :
06    <link href="css/bootstrap.min.css" rel="stylesheet">
      :
10    <main class="w-75 mt-5">
      :
12       <div class="d-flex">
      :
17           <h1 class="fs-2 border-bottom mb-4 pb-3 text-danger">
               イヌワシの生態<br>
               <span class="fw-normal fs-5 text-warning">Golden Eagle Ecology</span>
             </h1>
      :
24       <div class="d-flex mt-5 border p-4 bg-light">
      :
26           <h2 class="fs-2 border-bottom mb-4 pb-2 text-dark">
               猛禽類の定義の変遷<br>
               <span class="fw-normal fs-5 text-warning">Birds Of Prey</span>
             </h2>
      :
36    <script src="js/bootstrap.min.js"></script>
      :
```

問題02

lib_exercise_02.html

```
      :
10    <div class="w-75 d-flex mx-auto">
      :
37     <aside class="w-25 mt-5 ps-5">
38       <h2 class="fs-4 bg-black mb-0 p-3 text-center text-warning">その他の猛禽類</h2>
39       <ul style="list-style-type: none;padding-left: 0;"
             class="text-center p-3 text-secondary-emphasis bg-warning-subtle">
40         <li class="p-3 border-bottom"><a class="d-block" href="#">鷹 (Hawk) </a></li>
41         <li class="p-3 border-bottom"><a class="d-block" href="#">隼 (Falcon) </a></li>
42         <li class="p-3 border-bottom"><a class="d-block" href="#">梟 (Owl) </a></li>
43         <li class="p-3 border-bottom"><a class="d-block" href="#">百舌鳥 (Shrike) </a></li>
44         <li class="p-3"><a class="d-block" href="#">コンドル (Condor) </a></li>
      :
```

問題03

lib_exercise_03.html

```
      :
10    <div class="fixed-top w-100 bg-light">
      :
13          <ul style="list-style-type: none; padding-left: 0;"
                class="w-75 mx-auto pt-3 d-flex text-center text-secondary-emphasis">
14            <li class="w-25 p-2 border-end"><a class="d-block" href="#">TOP</a></li>
15            <li class="w-25 p-2 border-end"><a class="d-block" href="#">イヌワシの生態</a></li>
16            <li class="w-25 p-2 border-end"><a class="d-block" href="#">猛禽類の解説</a></li>
17            <li class="w-25 p-2"><a class="d-block" href="#">他の猛禽類</a></li>
      :
22    <div class="bg-black" style="padding-top: 80px;">
23      <img src="imgs/golden-eagle-top.jpg" alt="" class="w-100">
      :
63    <div class="bg-dark mt-5 p-3">
64      <footer class="fs-5 text-light">
65        <p class="text-center">
      :
```

問題04

lib_exercise_04.html

```
      :
63    <div class="d-flex w-75 mx-auto mt-5 pt-5">
      :
65        <img class="w-100" src="imgs/golden-eagle_1_330.jpg" alt="" data-bs-toggle="modal"
              data-bs-target="#modal_id_1">
      :
68        <img class="w-100" src="imgs/golden-eagle_2_330.jpg" alt="" data-bs-toggle="modal"
              data-bs-target="#modal_id_2">
      :
71        <img class="w-100" src="imgs/golden-eagle_3_330.jpg" alt="" data-bs-toggle="modal"
              data-bs-target="#modal_id_3">
      :
81    <div class="modal fade" id="modal_id_1" tabindex="-1" aria-hidden="true">
82      <div class="modal-dialog modal-dialog-centered modal-xl">
83        <img class="w-100" src="imgs/golden-eagle_1_1000.jpg" alt="">
      :
86    <div class="modal fade" id="modal_id_2" tabindex="-1" aria-hidden="true">
87      <div class="modal-dialog modal-dialog-centered modal-xl">
88        <img class="w-100" src="imgs/golden-eagle_2_1000.jpg" alt="">
      :
91    <div class="modal fade" id="modal_id_3" tabindex="-1" aria-hidden="true">
92      <div class="modal-dialog modal-dialog-centered modal-xl">
93        <img class="w-100" src="imgs/golden-eagle_3_1000.jpg" alt="">
      :
```

問題05

lib_exercise_05.html

```
      :
06    <link href="css/bootstrap.min.css" rel="stylesheet" type="text/css">
      :
14          <div class="container text-center mb-5" id="info" style="padding-top: 150px;">
      :
17            <div class="row w-75 mx-auto mt-5">
18              <div class="col-4">
      :
20                <p class="fs-5 text-info-emphasis text-start mt-4">“ペットを … </p>
21                <p class="p-2 mt-3 text-primary bg-primary-subtle mx-auto w-50 rounded-pill">
      :
23              <div class="col-4">
      :
25                <p class="fs-5 text-info-emphasis text-start mt-4">ブリーダーの … </p>
26                <p class="p-2 mt-3 text-primary bg-primary-subtle mx-auto w-50 rounded-pill">
      :
28              <div class="col-4">
      :
30                <p class="fs-5 text-info-emphasis text-start mt-4">販売すれば良い … </p>
31                <p class="p-2 mt-3 text-primary bg-primary-subtle mx-auto w-50 rounded-pill">
      :
38    <script src="js/bootstrap.min.js"></script>
      :
```

問題06

lib_exercise_06.html

```
      :
38          <h2 class="text-primary-emphasis fw-normal mx-auto p-2 border-top border-bottom
                    border-primary-subtle border-2 w-50">
              ペット用品
            </h2>
      :
40          <div class="d-flex w-75 mx-auto">
41            <div class="mt-4 px-4 w-50">
      :
43              <p class="border border-dark border-top-0">
                  <img src="imgs/pet_food.png" alt="" class="w-100">
                </p>
      :
46            <div class="mt-4 px-4 w-50">
      :
48              <p class="border border-dark border-top-0">
                  <img src="imgs/pet_toys.png" alt=""  class="w-100">
                </p>
      :
51          </div>
      :
```

問題07

lib_exercise_07.html

```
      :
58          <div class="row">
59            <div class="col card bg-body-tertiary p-2">
60              <div class="card-img-top"><img src="imgs/dogs-bulldog.png" class="w-100"></div>
61              <div class="card-body">
62                <h3 class="card-title fs-5 p-2 border-top border-bottom">ブルドッグ</h3>
63                <p class="card-text text-info-emphasis fw-light pb-2">2023年3月生まれ　牡</p>
      :
67            <div class="offset-1 col card bg-body-tertiary p-2">
68              <div class="card-img-top"><img src="imgs/dogs-beegle.png" class="w-100"></div>
69                <div class="card-body">
70                <h3 class="card-title fs-5 p-2 border-top border-bottom">ビーグル</h3>
71                <p class="card-text text-info-emphasis fw-light pb-2">2023年5月生まれ　牝</p>
      :
75          </div>
      :
```

問題08

lib_exercise_08.html

```
      :
64                <button type="button" class="btn btn-primary" data-bs-toggle="modal"
                          data-bs-target="#modalLabel1">
                    解説はこちら
                  </button>
      :
72                <button type="button" class="btn btn-primary" data-bs-toggle="modal"
                          data-bs-target="#modalLabel2">
                    解説はこちら
                  </button>
      :
78      <div class="modal fade" id="modalLabel1" tabindex="-1" aria-labelledby="modalLabel1"
            aria-hidden="true" data-bs-backdrop="static" data-bs-keyboard="false">
79        <div class="modal-dialog modal-dialog-centered text-center modal-dialog-scrollable">
80          <div class="modal-content p-3">
81            <div class="modal-body">
      :
87            </div>
88            <div class="modal-footer">
      :
90            </div>
      :
94      <div class="modal fade" id="modalLabel2" tabindex="-1" aria-labelledby="modalLabel2"
            aria-hidden="true" data-bs-backdrop="static" data-bs-keyboard="false">
95        <div class="modal-dialog modal-dialog-centered text-center modal-dialog-scrollable">
96          <div class="modal-content p-3">
97            <div class="modal-body">
      :
103           </div>
104           <div class="modal-footer">
      :
106           </div>
      :
```

問題09

lib_exercise_09.html

```
    :
12    <header>
13      <div class="container fixed-top bg-black opacity-75 w-75">
14        <div class="row">
15          <h1 class="col-5"><img src="imgs/logo.png"></h1>
16          <nav class="col-7 text-center">
17            <ul class="row text-light pt-4" style="list-style-type: none; padding-left: 0;">
18              <li class="col-3"><a href="#info">当店のご紹介</a></li>
19              <li class="col-3"><a href="#item">ペット用品</a></li>
20              <li class="col-3"><a href="#breeder">ペット販売</a></li>
21              <li class="col-3"><a href="lib_exercise_10.html">会員登録</a></li>
    :
24          </div>
    :
26      <p><img src="imgs/top_img.png" alt="" class="w-100"></p>
27    </header>
    :
127   <footer>
128     <div class="bg-dark text-center mt-4 p-4">
    :
131   </footer>
    :
```

問題10

lib_exercise_10.html

```
      :
12      <form class="needs-validation" action="" method="post" novalidate>
13        <div class="row mb-3">
14          <label class="col-2 col-form-label" for="name">氏名</label>
15          <div class="col-10">
16            <input class="form-control" type="text" id="name" name="name"
                    placeholder="氏名を入力" pattern="…" required>
17            <div class="invalid-feedback">氏名は必須項目です。</div>
18          </div>
19        </div>
20        <div class="row mb-3">
21          <label class="col-2 col-form-label" for="tel">電話番号</label>
22          <div class="col-10">
23            <input class="form-control" type="tel" id="tel" name="tel"
                    placeholder="電話番号をハイフンなしで入力" pattern="…" required>
24            <div class="invalid-feedback">電話番号は必須項目です。</div>
25          </div>
26        </div>
27        <div class="row mb-3">
28          <label class="col-2 col-form-label" for="address">住所</label>
29          <div class="col-10">
30            <input class="form-control" type="text" id="address" name="address"
                    placeholder="住所を入力" pattern="…" required>
31            <div class="invalid-feedback">住所は必須項目です。</div>
32          </div>
33        </div>
34        <div class="row mb-3">
35          <label class="col-2 col-form-label" for="email">メールアドレス</label>
36          <div class="col-10">
37            <input class="form-control" type="email" id="email" name="email"
                    placeholder="メールアドレスを入力" pattern="…" required>
38            <div class="invalid-feedback">メールアドレスは必須項目です。</div>
39          </div>
40        </div>
41        <hr>
42        <input type="submit" value="送信する" class="btn btn-primary rounded-pill"
                  style="width: 20%;">
43        <input type="reset" value="リセット" class="btn btn-secondary rounded-pill"
                  style="width: 20%;">
44      </form>
      :
```

令和５年度「専修学校による地域産業中核的人材養成事業」
IT 分野 DX 人材養成のモデルプログラム開発と実証事業

フロントエンドエンジニア教材資料 2

令和 6 年 2 月

一般社団法人全国専門学校情報教育協会
〒164-0003　東京都中野区東中野 1-57-8　辻沢ビル 3F
電話：03-5332-5081　FAX　03-5332-5083