# やってみよう！
# Web プログラミング

内　容

- HTML/CSS 入門
- JavaScript 入門
- 課題プログラムの作成

# 目次

# HTML/CSS 入門

## 1. 前提知識と事前準備

### 1.1. Web とは

HTML、CSS、JavaScript は Web サイトを作るための技術の一部です。Web とは World Wide Web の省略形で、インターネットを使った世界規模のクモの巣を表します。Web サイトはブラウザで閲覧できるホームページの別の言い方です。クモの巣のように、ページ同士がつながっているイメージです。

ブラウザはアプリの一種で、Web サイトを閲覧するためのソフトウェアです。クローム、サファリ、ファイヤーフォックス、エッジなどがあります。Windows には最初からエッジとインターネット・エクスプローラが入っています。Mac や iPad・iPhone にはサファリが入っています。図 1 はブラウザのシェア（どれくらい使われているか）です。

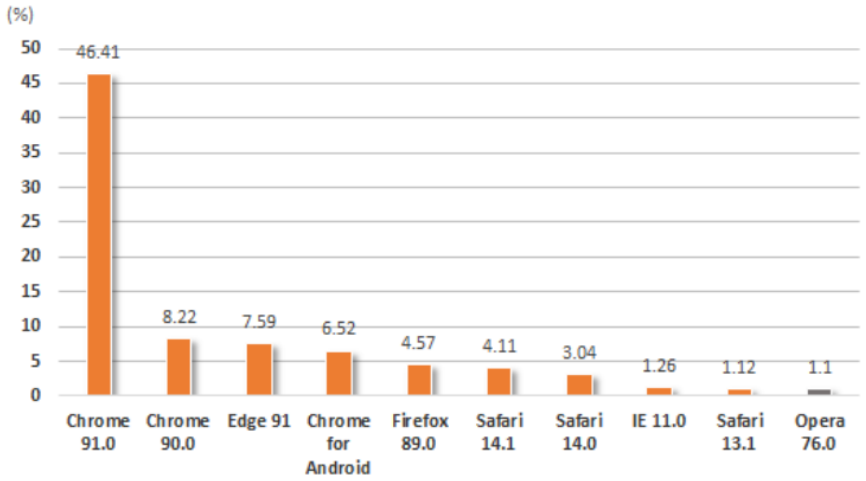



**図 1　ブラウザの世界シェア（2021 年 7 月 スタートカウンター社調べ）**

主要なブラウザのアイコンを図 2 に示します。左から、ファイヤーフォックス、クローム、インターネット・エクスプローラ、エッジです。インターネット・エクスプローラは開発が終了しており、セキュリティの観点から使わないように強く推奨されています。


**図 2　ブラウザの種類とアイコン**

HTML、CSS、JavaScript はそれぞれ、Web サイトを作るための技術です。HTML はコンテンツの表示と基本的な見た目を決定します。CSS は見た目をより良くします。JavaScript はプログラミング言語の一種で、動きをつけたり、業務処理を行います。

### 1.2. Web システムの構成図

システムを理解するには、構成図を描くとわかりやすいです。図 3 は一番シンプルな Web システムの構成図です。

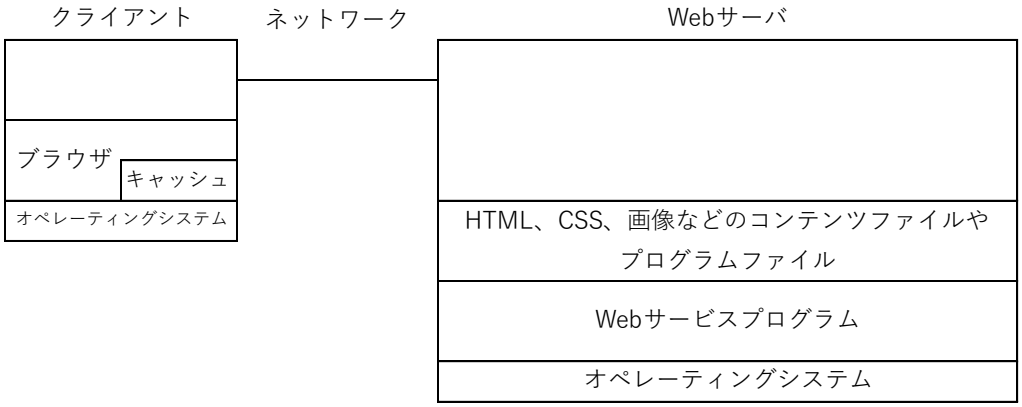



**図 3　Web システムの構成図**

処理手順とデータの流れについて説明します。

- Web サーバとクライアントがネットワークで接続されています。
- どちらのコンピュータでもオペレーティングシステム（OS）が動いています。OS がなければ、コンピュータは動くことができません。
- クライアントのブラウザが Web サーバの Web サービスプログラムに「ファイルをくれ」と要求を出します。
- Web サービスは要求に応え、指定されたファイルをブラウザに送ります。
- ブラウザは取得したファイルを画面に表示します。
- ブラウザが受け取ったファイルは一旦キャッシュに溜めておきます。次に同じファイルを要求しようとした場合、Web サーバに問い合わせる前にキャッシュを探します。

### 1.3.　プログラミングをする上での最低限の環境設定

HTML、CSS、JavaScript はファイルとして作成します。ファイルの名前には「拡張子」というものがあります。プログラミングするうえで、ファイルの拡張子が見えていた方がいいので、Windows の方は図 4 のように「ファイル名拡張子」をチェックしてください。

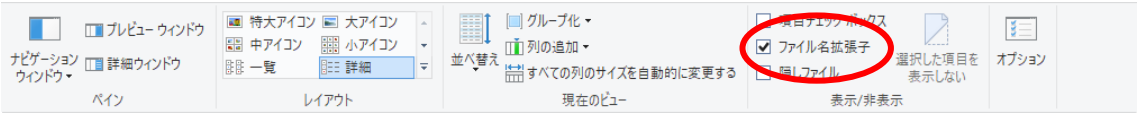



**図 4　エクスプローラの表示メニューで「ファイル名拡張子」をチェック**

Mac の方は、Finder の環境設定＞詳細＞すべてのファイル名拡張子を表示　です。


**図 5　Mac での設定**

### 1.4.　半角と全角、小文字と大文字の違い

コンピュータで入出力する文字には大きく分けて半角・全角の違い、小文字・大文字の違いがあります。プログラムを作るとき、これを間違うとエラーになります。**プログラムは半角で命令を書きます**。

まず、半角と全角の違いについてです。漢字変換がかかわってきます。漢字キーを押すと漢字変換モードになり、入力する文字が全角になります。もう一度漢字キーを押すと半角に戻ります。「A」は半角の A、「Ａ」は全角の A です。全角の方が少し横長になります。困るのは、スペース（空白）を入出力したときです。「 」は半角のスペース、「　」は全角のスペースです。全角のスペースが半角より横長なのは確かですが、「　」が全角スペースなのか半角スペース 2 つ分なのか、Windows のメモ帳（ちょう）、Mac のテキストエディットでは区別がつきにくいです。そこで、プログラミングする場合は、プログラミング用のエディタをインストールするのが普通です。図 6 はプログラマがよく使うエディタ「サクラエディタ」で A とスペースを半角・全角で入力してみたものです。全角スペースは「□」のように表示されます。また、縦方向に行数、横方向に桁数（列数）が表示されるので、何行・何文字入力したかわかりやすくなっています。

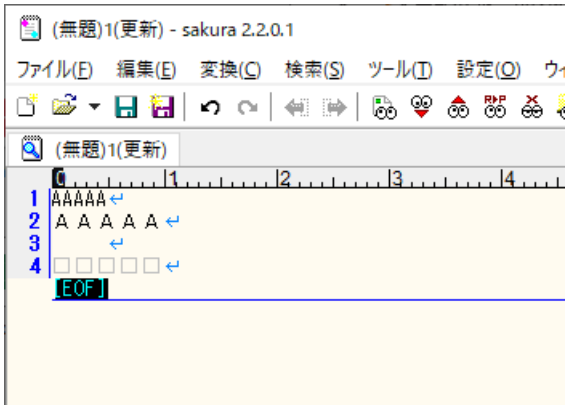



**図 6　サクラエディタで文字を入力したところ**

本テキストでは、エディタのインストールについて解説しません。可能でしたら、ブラウザでインターネットを検索し、良さそうなエディタを選んでインストールしてみてください。インストールしない人は、Windows ではメモ帳、Mac ではテキストエディットを使ってみましょう。これ以降、Windows を使う前提で解説します。

小文字・大文字の違いについても補足します。英語には小文字と大文字があります。「A」は大文字、「a」は小文字です。多くのプログラミング言語は、小文字と大文字を区別します。HTML と CSS はあまり区別しません。JavaScript は完全に区別します。

### 1.5. ファイルの文字コード

テキストファイル（プログラムで言えばソースコード）を保存するときは、文字コードを気にする必要があります。文字コードにはいくつも種類があります。Windows は長い歴史の中で「Shift_JIS（ANSI）」を使ってきました。Mac は UTF-8 です。あるシステムの中で、このプログラムは ANSI、このプログラムは UTF-8、などとなるとバグのもとになります。1 システム内の文字コードは統一しなければなりません。

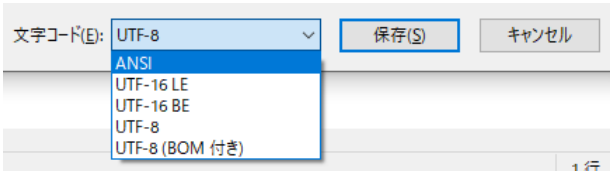



**図 7　Windows のメモ帳の文字コード指定**

Web システムでも同様で、ブラウザが期待している文字コードと実際の HTML の文字コードに差異があると文字化けが発生します。これについては後述します。

## 1.6.　本テキストで使用するファイル

本テキストで使用するファイルを、あらかじめ配布します。作業用のフォルダを作って、そのフォルダにファイルを保存してください。例えば、デスクトップに「html」というフォルダを作り、配布されたファイルをその html フォルダに移します。そうすると、デスクトップの html アイコンをダブルクリックすることで、そのフォルダをエクスプローラで開くことができます。

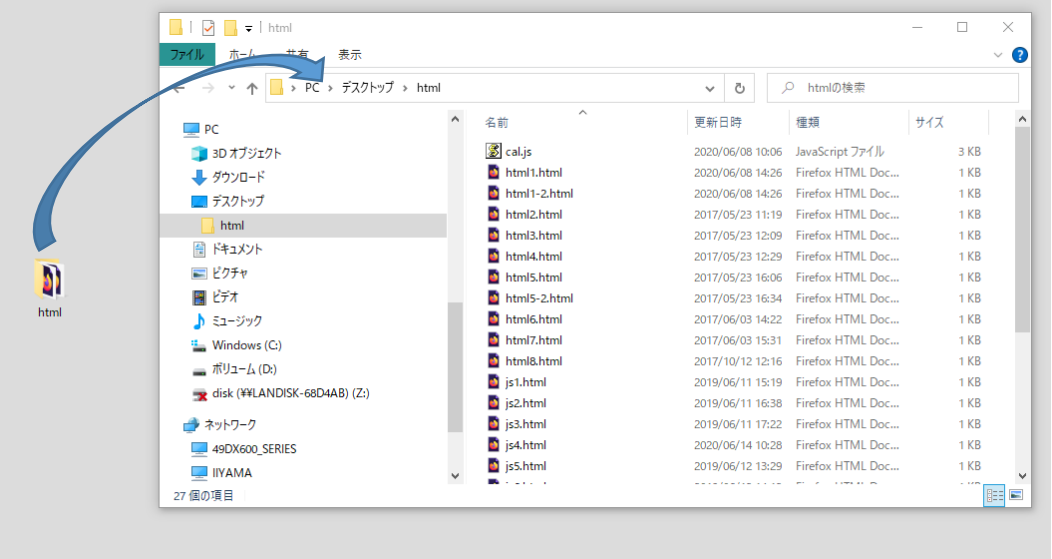



**図 8　デスクトップにフォルダを作る例**

## 2　HTML ファイルの構成

本章から、まずは HTML について解説します。HTML ファイルは以下の構成が基本となります。

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>文書のタイトル</title>
</head>
<body>
文書の本文
</body>
</html>
```

**図 9　HTML 基本形**

- 小なり大なり記号「<>」の中の「html」や「body」が***タグ***と呼ばれるキーワードになります。これらのタグは大文字・小文字どちらでも OK ですが、このテキストでは小文字で書きます。
- 一番先に来るのが<!doctype html>です。スペースは半角で入力してください。
- <html>は HTML の開始を表し、</html>は終了を表します。対(ついになっています。
- <head>はヘッダ部の開始を表し、</head>は終了を表します。対になっています。
- <body>は本文部の開始を表し、</body>は終了を表します。対になっています。
- このように、タグは開始と終了を明示する形となります。終了を忘れることが多いので気をつける必要があります。終了を忘れると、画面が崩れたり期待通りの表示にならなくなります。
- <!doctype html>のように開始と終了がなく単体で使うものも少しあります。
- meta タグで文字コードを指定すると、文字化けを抑止できます。しかし、meta タグで指定した文字コードと、実際のファイルの文字コードが異なると図 10 のように文字化けしてしまいます。


**図 10　meta タグで指定した文字コードが間違っている**

[練習]

1. エディタまたはメモ帳を起動してください。メモ帳は、Windows メニューの「Windows アクセサリ」の中にあります。

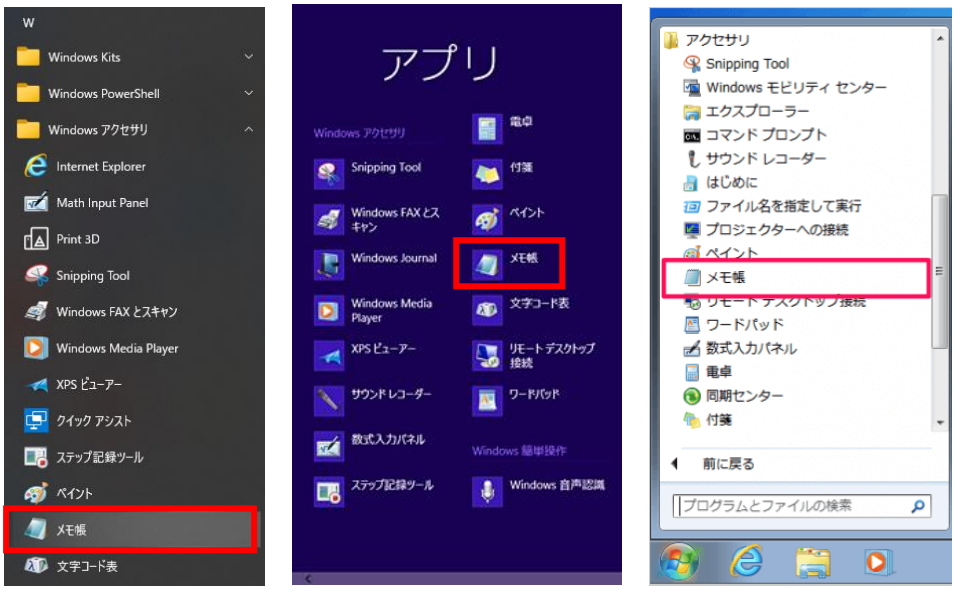



図 11　Windows メニュー（左から Win10, Win8.1, Win7）

2. HTML ファイルを作りましょう。図 9 を真似して、エディタまたはメモ帳でタイトルと本文に適当な文字列を書き、html1.html という名前をつけて保存してください。
3. 作ったファイルをブラウザで読み込んでください。

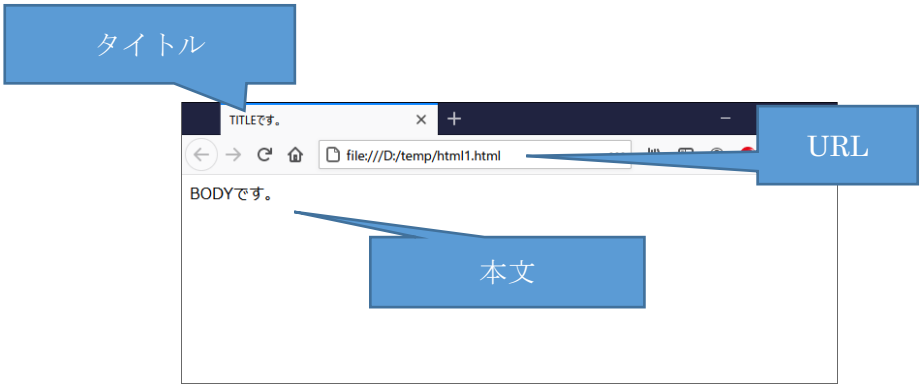



図 12　このように表示されましたか？

ブラウザに HTML ファイルを読み込ませるには、

- エクスプローラで HTML ファイルをつかみ、ブラウザに落とす（ドラッグ・アンド・ドロップする）方法
- HTML ファイルを右クリックし「開く」を選択する方法
- HTML ファイルをダブルクリックする方法

があります。

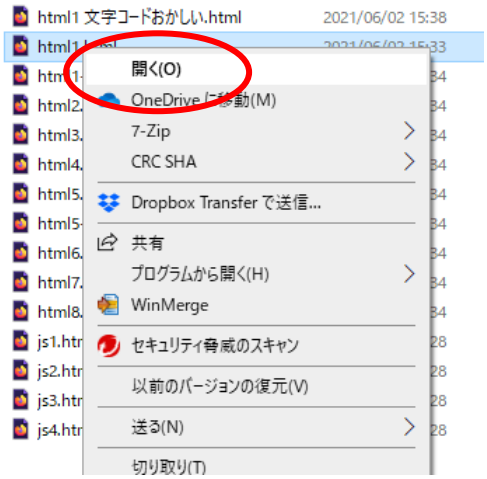



**図 13　Windows で、ｴｸｽﾌﾟﾛｰﾗで HTML ﾌｧｲﾙを右ｸﾘｯｸして表示されるﾒﾆｭｰ**

エディタやメモ帳に HTML ファイルを読み込ませるには、

- エクスプローラで HTML ファイルをつかみ、エディタやメモ帳に落とす（ドラッグ・アンド・ドロップする）方法
- エディタやメモ帳でファイルを「開く...」からファイルを指定する方法

があります。

ここで、ブラウザとエディタ・メモ帳の使い方に慣(な)れてください。

## 3　見出し

```
<h1>見出し 1</h1>
<h2>見出し 2</h2>
<h3>見出し 3</h3>
<h4>見出し 4</h4>
<h5>見出し 5</h5>
<h6>見出し 6</h6>
```

図 14　見出し

見出しは新聞の見出しのように、書く内容の題名や概要を表します。タグは h1～h6 まであり、数が小さい方が見出しが大きくなります。図 14 は!doctype html や html タグは省略し、body タグの中だけを書いています。これ以降も同様とします。

[練習]

html1.html を流用し、本文部に上記見出し 1～6 を書いて、html2.html として保存してください。ブラウザで開いて、見出しの違いを確認してください。

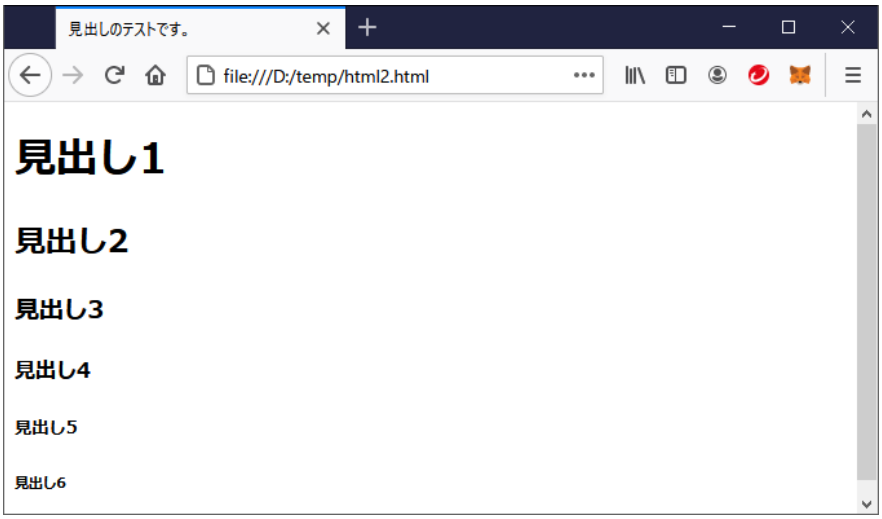



図 15　見出し 1～6 の違い

## 4　段落と改行

```
<h1>当社の歴史</h1>
<p>
　当社は 2015 年 4 月に新潟県長岡市であいうえおかきくけこさしすせそたちつてとなにぬねのはひふへほまみむめもやゐゆゑよらりるれろわをん。<br>
　創業者のあいうえおかきくけこさしすせそたちつてとなにぬねのはひふへほ。
</p>
<p>
　当社の経営理念はあいうえおかきくけこさしすせそたちつてとなにぬねのはひふへほまみむめもやゐゆゑよらりるれろわをん。
</p>
```

図 16　段落と改行

HTML は編集時に改行してもブラウザで表示したときには改行されません。改行するには、改行を指示するタグを指定する必要があります。タグ p で囲まれた中が一つの段落となり、段落終了時に少し高さを持った改行が行われます。(行間が少し空く)

段落内で強制的に改行したい場合は、終わりのないタグ br を使います。(行間はほとんど空かない)


図 17　段落改行と強制改行の違い

[練習]

p と br を使った Web ページを作ってください。書く内容はなんでもかまいません。ファイル名は html3.html にしてください。

## 5　箇条書き

```
<p>
持ってくるもの
<ul>
<li>水筒</li>
<li>おやつ（300 円まで）</li>
<li>ハンカチ、ポケットティッシュ</li>
</ul>
</p>
<p>
当日の行動
<ol>
<li>駅に 8:00 に集合</li>
<li>点呼</li>
<li>8:20 の〇〇行きに乗車</li>
<li>10:03 に〇〇到着</li>
</ol>
</p>
```

**図 18　箇条書き**

HTML で書ける箇条書きは標準で二種類あります。黒丸●と数字によるものです。黒丸●は ul タグを使い、数字は ol タグを使います。列挙[1]する項目は li で書きます。

[練習]

ul と ol を使って何か箇条書きにしてください。ファイル名は html4.html です。

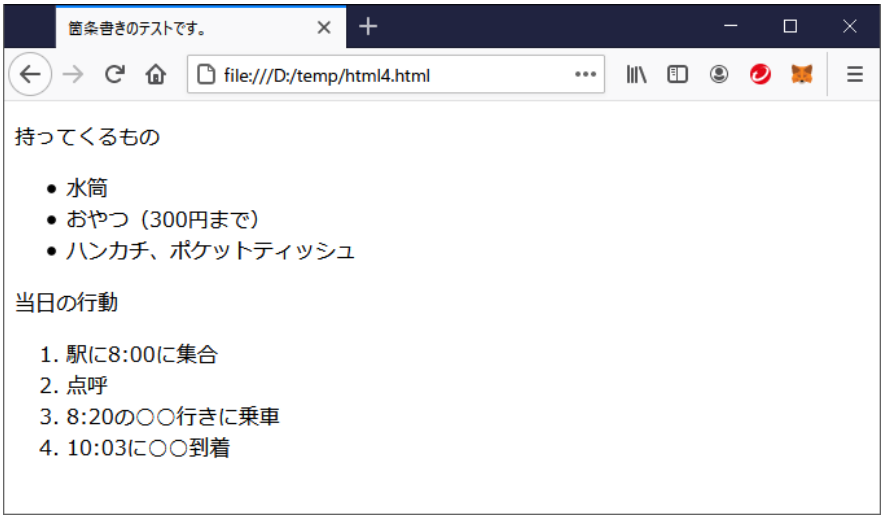



**図 19　二種類の箇条書き**

[1] 一つ一つ数えて並べることを列挙といいます。

## 6　表

```
<table align="center">
<caption>表 1 売上個数推移</caption>
<tr><th>年月</th><th>りんご</th><th>みかん</th></tr>
<tr><td>2016/7</td><td align="right">43</td><td align="right">33</td></tr>
<tr><td>2016/8</td><td align="right">52</td><td align="right">29</td></tr>
<tr><td>2016/9</td><td align="right">55</td><td align="right">24</td></tr>
</table>
```

図 20　表

表は行と列を持ったデータの集まりです。表を作るのは table タグです。caption タグを使えば、表に説明をつけられます。tr タグ 1 つで 1 行分のデータを作ります。th タグは行の先頭がデータ名のときに使います。th を使うと、行の先頭が強調されます。th は使わなくてもかまいません。

td タグはデータを 1 つ持ちます。列の分だけ td を書きます。tr、th、td は閉じるのを忘れると表が崩れたり期待通りの表示にならないので注意してください。

align パラメータは表示位置の属性です。table に align="center"をつけると、画面の中央に表を表示します。図 20 ではりんごとみかんという数を扱っているため、りんごの列とみかんの列に align="right"として右寄せの属性を付与しました。

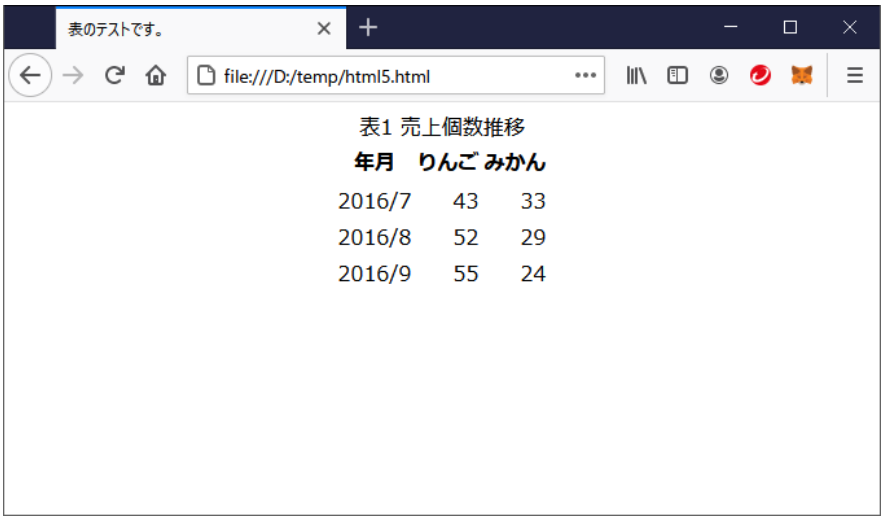



図 21　図 20 をブラウザで表示したもの(html5.html)

[小テスト 1]

図 21 は表に枠線（罫線）がありません。table タグの仕様を調べて、枠線をつけてください。ファイル名は html5-2.html にしてください。

[PC でネットの情報を調べる方法]

ブラウザで「検索サイト」を開きます。ここでは、Google 検索を開きます。ブラウザのアドレス欄に「https://www.google.co.jp/」と入力し Enter キーを押します。あるいは、ブラウザによっては、アドレス欄自体が検索キーワードを入力できるようになっているものもあります。


図 22　ブラウザで Google 検索を開いたところ

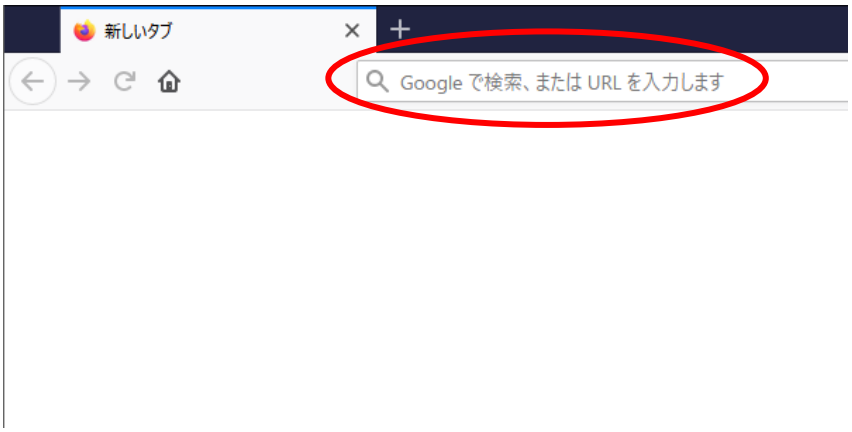



図 23　アドレス欄に検索キーワードを入れられるブラウザもある

HTML の table タグの枠線（罫線）について知りたいので、キーワードごとにスペースをあけて「html table 枠線」と入力します。検索時は大文字小文字の区別はありません。すると、検索候補がいくつか出てきます（図 24）。これは、他の人もこのようなキーワードで検索サイトを調べていることをあらわします。


**図 24　検索候補**

検索結果は図 25 のように表示されます。良さそうな検索結果が得られました。ここから答えを見つけてください。気に入ったサイトはブックマーク（しおり）をつけて、何度でも訪問するといいでしょう。

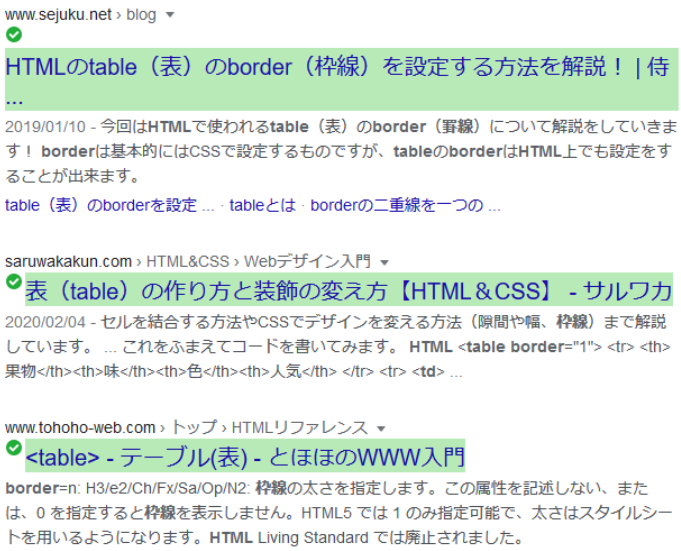



**図 25　検索結果（一部）**

これ以降、わからないところが出てきたら、検索して調べてください。

## 7　ハイパーリンク

HTML 文書の特徴の一つとして、他の HTML 文書への結びつきを作られることがあげられます。結びつきのことをハイパーリンク（リンク）と呼び、この文書からあの文書へリンクをたどって飛ぶことができます。ハイパーリンクは a タグを使い、href パラメータに URL を指定します。

図 26 はリンクの例です（hyperlink.html）。ここまでで作った html1.html～html4.html へ次のようにリンクしています。

```
<ul>
<li><a href="./html1.html">HTML の基本</a></li>
<li><a href="./html2.html">見出し</a></li>
<li><a href="./html3.html">段落と改行</a></li>
<li><a href="./html4.html">箇条書き</a></li>
</ul>
```

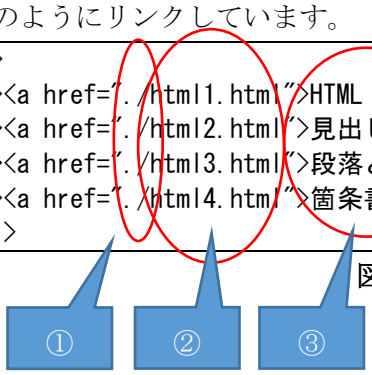



**図 26　ハイパーリンク**

①ドット「.」はカレント[2]フォルダを表します。hyperlink.html があるフォルダがカレントです。スラッシュ「/」はフォルダの区切りです。
②飛び先のファイルです。
③飛び先の内容を表します。

[2] カレント（current）は現在のという意味です。

hyperlink.html を表示すると以下のようになります。リンク「HTML の基本」をクリックすると、html1.html に飛びます。

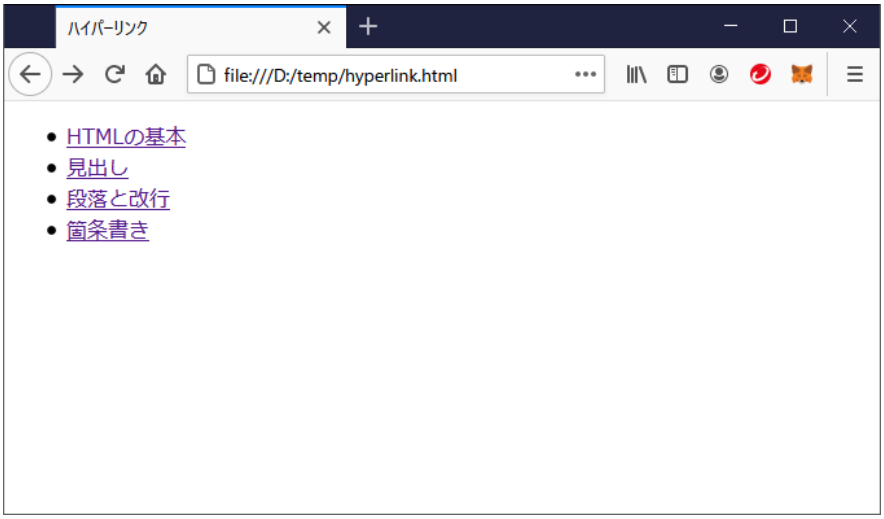



**図 27　ハイパーリンクの表示**

a タグに指定するのは、ローカル[3]にあるファイルでなくてもかまいません。

```
<a href= "https://www.asahi.com/">朝日新聞</a>
```

とすれば、朝日新聞のページに飛びます。

[3] ローカル（local）は自分のコンピュータという意味です。

# 8　画像を表示する

## 8.1.　HTML での画像の扱い方

img タグを使うと、ブラウザ上に画像を表示することができます。src（ソース）パラメータにファイル名を指定します。ハイパーリンクと同様に、ファイルはローカルでもいいしリモート[4]でもいいです。図 28 は画像表示の例です。image というフォルダの中に画像ファイル「テスト画像.jpg」が入っています。

```
<img src="image/テスト画像.jpg" alt="テスト画像">
```

**図 28　テスト画像.jpg という画像を表示する**

src パラメータに画像ファイルのパスまたは URL、alt（オルト）には代替文字列を指定します。代替文字列とは、画像を表示するかわりに表示する文字列のことです。ネットワークが遅すぎるために画像が見られない環境で使っている人や視覚障碍（がい）者のために、文字で画像の内容を説明します。img タグは終了がありません。

[小テスト 2]

a タグと img タグを組み合わせて、リンクではなく画像をクリックすると指定の URL に飛ぶようにすることができます。

検索サイト「Google」のロゴ画像をクリックすると、Google のホームページに飛ぶように HTML を書いてください。HTML ファイル名は html6.html としてください。

Google ホームページ：https://www.google.co.jp/

Google のロゴ画像：https://www.google.co.jp/images/branding/googlelogo/1x/googlelogo_color_272x92dp.png

（Google サイトでロゴ画像を右クリックして「画像の URL をコピー[5]」すれば URL を取得できます）


**図 29　Google ロゴ画像の URL を得る方法**

---

4 リモート（remote）は遠隔地にあるコンピュータという意味です。

5 ブラウザによっては「画像のリンクをコピー」「画像アドレスをコピー」となります。

## 8.2. 画像の作り方

Windows では、「ペイント」で画像を作ることができます。Web サイトの場合、画像の種類は以下を使用します。慣習的に、BMP は使用しません。

- JPEG（一般的にはこれ。拡張子は.jpg または.jpeg）
- PNG（透過させたい場合はこれ。拡張子は.png）
- GIF（アニメを使いたいならこれ。拡張子は.gif）

今、コンピュータの画面に映っている画像を取得したいときがあります。3 つの操作方法で、今映っている画像をファイルにすることができます。これを「画像のキャプチャ」といいます。

[方法 1 全体画像]

1. Print Screen キーを押します[6]。クリップボードに全体画像が保持されます。
2. ペイントを起動します。
3. 貼り付け(Ctrl+v)で画像を貼り付けます。
4. 画像が貼り付いたら、好きな種類で画像をファイルとして保存します。

[方法 2 ウィンドウ単位画像]

1. キャプチャしたいウィンドウをクリックしアクティブにします。
2. Alt+Print Screen キーを押します。クリップボードにウィンドウ画像が保持されます。
3. これ以降は方法 1 と同じです。

[方法 3 矩形画像]

矩形画像とは、画面の一部を切り取った画像です。例えば、何かの説明書を書くことを考えましょう。「このボタンを押すと、何何機能が起動します。」と書いたとすると、そのボタンの画像だけが欲しくなります。矩形画像のキャプチャは⊞+Shift+s で撮れます。

[練習]

上記の方法で画像ファイルを作りましょう。image.html を作り、キャプチャした 3 つの画像を表示させてみてください。

[6] PrintScreen は、キーボードによっては別の何かのキーと一緒に押す必要があります。

## 9　フォーム

form タグを使って対話的な画面を作ることができます。画面の中に入力項目、ボタン、ドロップダウンリストなどを配置することができます。

```
<form>
各画面の項目...
</form>
```

図 30　フォームタグ

フォームと項目で作った画面の例を図 31 に示します。

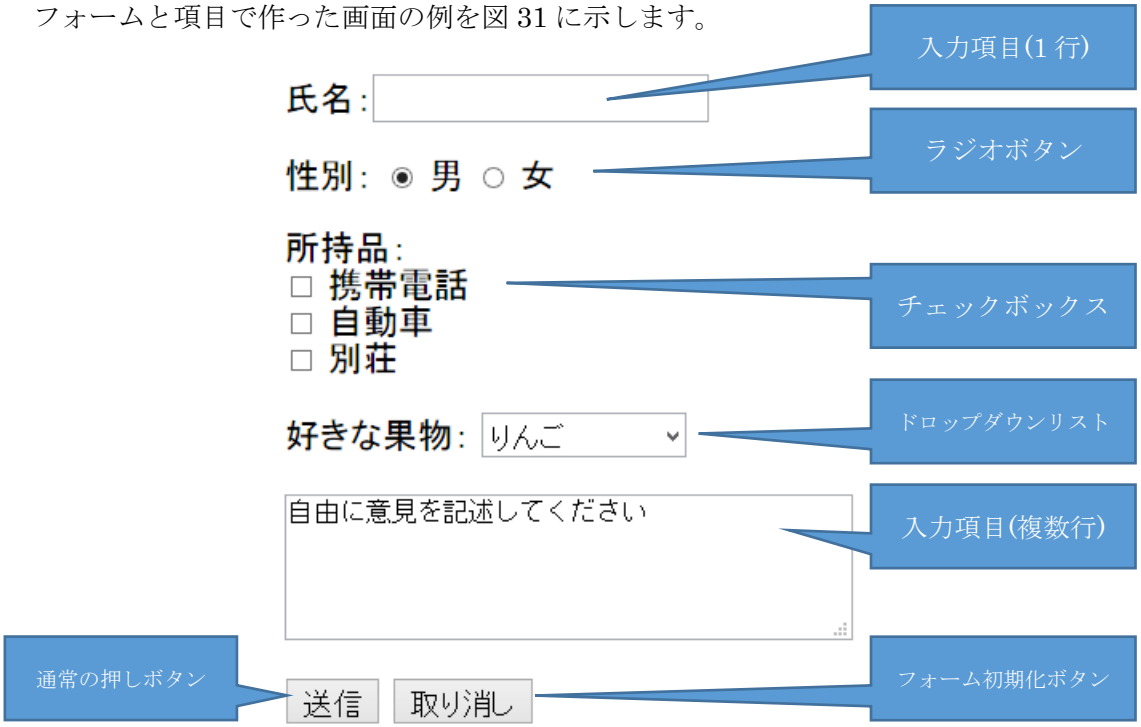



図 31　フォームで作った画面

これから、フォーム内で作ることができる画面項目を一つ一つ説明します。

入力項目は input タグでパラメータ type を text にします。id というのは画面内の各項目が持つユニーク[7]な識別子です。

```
氏名：<input type="text" id="氏名">
```

図 32　入力項目（1 行）

複数の中からどれか 1 つを選ぶボタンをラジオボタンといいます。ラジオボタンは input タグでパラメータ type を radio にします。各ボタンは name パラメータで同じ名前を持たせます。デフォルト[8]でどれかを選択させたい場合、パラメータ checked に checked を指定します。

```
性別：
 <input type="radio" name="性別" value="男" checked="checked"> 男
 <input type="radio" name="性別" value="女"> 女
```

図 33　ラジオボタン

複数の中から 0 個以上を選ぶボタンをチェックボックスといいます。0 個以上なので、何も選ばなくてもいいし全部選んでもいいです。チェックボックスは input タグでパラメータ type を checkbox にします。ラジオボタンと同じく、各ボタンは同じ名前を持たせます。

```
所持品：<br>
 <input type="checkbox" name="所持品" value="携帯電話"> 携帯電話<br>
 <input type="checkbox" name="所持品" value="自動車"> 自動車<br>
 <input type="checkbox" name="所持品" value="別荘"> 別荘
```

図 34　チェックボックス

[7] ユニーク（unique）は世界でただ一つ、重複しないという意味です。例えば、メールアドレスが重複すると、メールを A さんに送ったつもりが B さんにも届いてしまいます。これではまずいので、メールアドレスはユニークである必要があります。ここでの意味は、1 つの HTML の中で重複しないということです。

[8] デフォルト（default）は標準、初期状態という意味です。後述するラジオボタンもチェックボタンもドロップダウンリストも、画面を表示したときにどれかを選んでいる（押されている）状態にすることができます。このとき、「デフォルトで○○が選ばれている」と言います。

複数の選択肢の中から 1 つを選ぶものとしてラジオボタンを説明しましたが、選択肢が増えるとラジオボタンでは扱いづらくなります。選択肢が多い場合にラジオボタンの代わりに使うのがドロップダウンリストです。ドロップダウンリストは select タグの中に各項目として option タグを書きます。

```
好きな果物：
 <select id="好きな果物">
   <option>りんご</option>
   <option>みかん</option>
   <option>バナナ</option>
   <option>パイナップル</option>
 </select>
```

図 35　ドロップダウンリスト

1 行の入力項目には input type="text"を使いますが、複数行は入力できません。複数行の入力には textarea タグを使います。textarea タグは rows パラメータで行の高さを、cols パラメータで列の幅を指定できます。

```
<textarea id="自由意見" rows="4" cols="40">
自由に意見を記述してください</textarea>
```

図 36　入力項目（複数行）

通常の押しボタンの場合、input タグで type は button にします。特殊なボタンもあります。type を reset にすると、フォーム全体をデフォルトに戻します。

```
<input type="button" value="送信">
<input type="reset" value="取り消し">
```

図 37　通常の押しボタンとフォーム初期化ボタン

ボタンが押されたときどうするか、各項目に何が入力されたのかを知るためには、JavaScript でのプログラミングが必要になります。それについては後述します。

[練習]

図 30～図 37 の内容を html7.html にまとめ、ブラウザに表示させて動作を確認してください。

## 10 カスケーディング・スタイル・シート（CSS）

HTML のタグの多くには、style 属性がつけられます。style 属性は、タグに見た目を指定するものです。html1.html をエディタで開いて、body タグに図 38 のように style 属性をつけてみてください。ファイル名は html1-2.html にして保存してください。

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>TITLE です。</title>
</head>
<body style="background: #00ff00;">
BODY です。
</body>
</html>
```

**図 38　style 属性を HTML のタグに追加する**

修正後の html1-2.html をブラウザで開くと、図 39 のように背景が緑色になります。

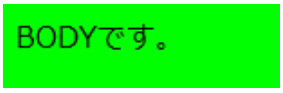



**図 39　実行結果**

上の例では背景を緑色にできましたが、一つ一つのタグに style をつけていくのはかなり手間です。そこで、手間を省くために CSS を使います。CSS は画面の見た目を良くするための style 設定が集まったものです。CSS を読み込むと、HTML 内にある各タグの表示方法が CSS 内で定義した方法に塗り替わります。

[CSS を使うメリット]

- 一つ一つの画面項目に設定をしなくてよくなる。
- 複数の画面があっても、同じ CSS を使えば画面に統一性を持たせられる。

style.css が同じフォルダにある状態で、html8.html をブラウザで開いてください。図 40 のように表示されます。これは、html7.html に CSS を指定したものです。背景画像が追加され、フォント（文字のデザイン）が変わったことも確認してください。

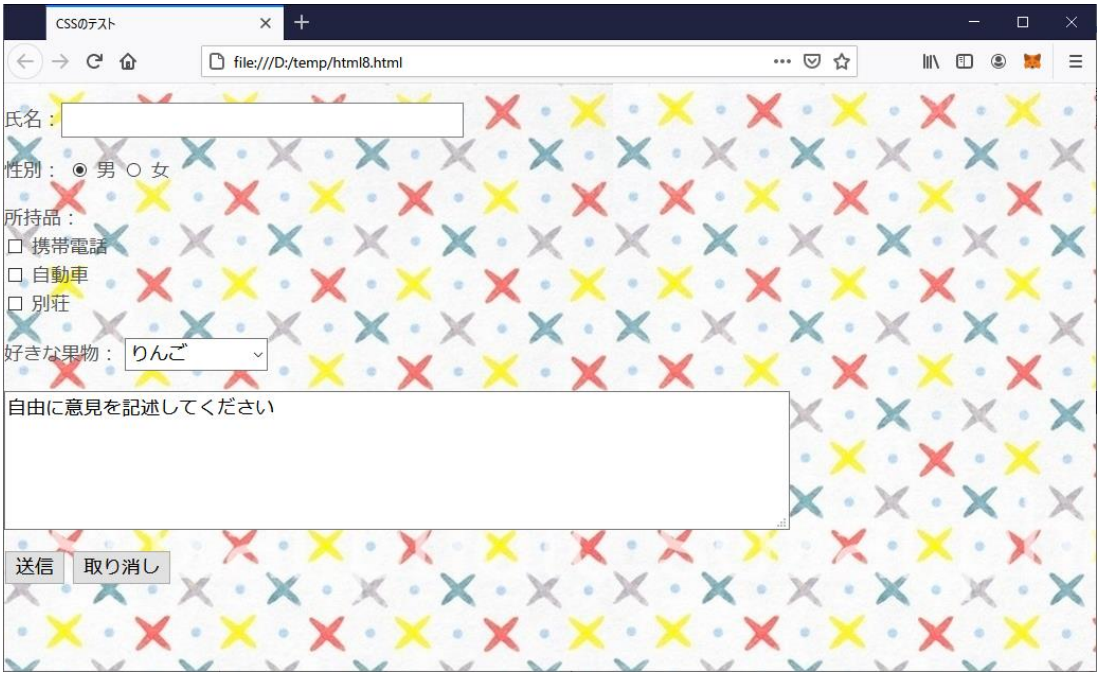



**図 40　CSS を指定し背景とフォントが変わった**

html8.html では、html7.html に次の 1 行を追加しました。HTML と同じカレントフォルダにある style.css を読み込みます。

```
<title>CSS のテスト</title>
<link rel="stylesheet" href="./style.css" type="text/css" />
```

この style.css に記述されている設定を見ていきましょう。なお、CSS の各種設定をおぼえる必要はありません。CSS でどんなことができるのかがわかればいいです。

CSS でも、HTML と同じように文字コードを指定します。また、文の最後にはセミコロン「;」をつけます。

```
@charset "UTF-8";
```

/* と */で囲まれた部分はコメントといいます。コメントはブラウザに解釈されない部分です。CSS の設定は人間にとってわかりにくいので、わかりやすい説明を書いておくのがコメントの意義です。

```
/* body タグの設定 */
```

style.css には、4 つのタグに対しての設定があります。body、input、select、そして textarea です。波括弧（なみかっこ）{}で囲んだ中に設定します。

```
body{
    body タグへの設定
}

input{
    input タグへの設定
}

select{
    select タグへの設定
}

textarea{
    textarea タグへの設定
}
```

border は枠線の設定です。0、solid、dashed などを設定します。図 41 は dashed red（赤い点線）を指定した例です。

```
    border:0;
```

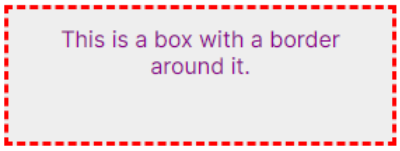



**図 41　border: dashed red; の例**

margin は要素の外側の余白の設定です。余白のサイズを設定します。

```
    margin:0;
```

padding は要素の内側の余白の設定です。余白のサイズを設定します。

```
    padding:0;
```

border、margin、padding の関係を整理すると、図 42 のようになります。

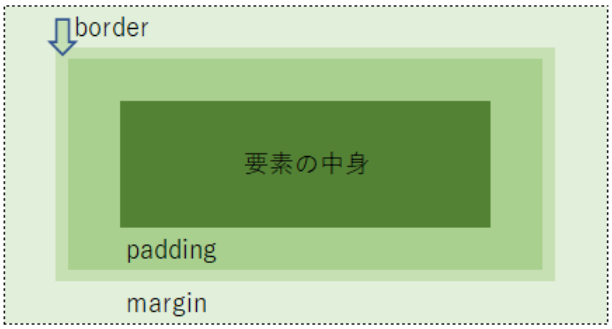



**図 42　border、margin、padding の関係**

font-size はフォントサイズの設定です。

```
    font-size:18px;
```

font-family はフォントの設定です。Web ページの閲覧者の環境（OS やブラウザ）によって使えないフォントがあるので、優先順にフォント名を列挙します。

```
    font-family:'meiryo','メイリオ','ms pgothic','ms p ゴシック',osaka,sans-serif;
```

color は文字色の設定です。あらかじめ定義されている red や blue という名前を指定してもいいですし、#00ff00 のように 16 進数で RGB[9]を指定してもいいです。

```
    color:#555555;
```

min-width は幅の設定です。min-width とは別に width で幅を設定することもできますが、width では min-width で決められた幅より小さくすることはできません。

```
    min-width:1020px;
```

background は背景の色を設定したり、背景画像を読み込みます。

```
    background:transparent url(image/kikagaku.jpg);
```

background-attachment は背景画像のスクロール設定です。scroll、fixed などを設定します。

```
    background-attachment: fixed;
```

タグに style 属性をつける方法と CSS 読み込みは同時に併用できますが、タグへの個別の style 属性付与の方が CSS より優先されます。

CSS には他にも細かい各種設定項目がたくさんあり、説明しきれません。「画面のここをこうしたい」という要望から、実現方法を調べていくといいでしょう。

[練習]

style.css の設定を変えて、画面表示がどう変わるか見てください。

[9] 赤:Red、緑:Green、青:Blue を光の三原色といいます。

# JavaScript 入門

## 11. プログラミング言語の基本形

プログラムは、いろいろな命令の集まりです。プログラミング言語にはたくさんの種類があり、JavaScript は主に Web システムに使われる言語です。どのようなプログラミング言語にも、以下の 4 つの基本形があります。

- 順次
- 比較（分岐）
- 繰り返し
- 抽象化（ちゅうしょうか）

JavaScript にもこれらの基本形の書き方があります。JavaScript でそれぞれどのように書くのか、見ていきましょう。

## 12. 順次

### 12.1. 順次の概要

1 つ目の基本形は「順次」です。プログラムの命令は、上から下へ、1 つ 1 つ順番に流れていきます。図 43 は順次のフローチャート（流れ図）の例です。

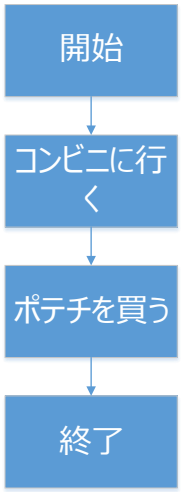



**図 43　順次のフローチャート**

順次の JavaScript プログラムの例を図 44 に示します。(js1.html)

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>順次</title>
</head>
<body>
<script>
document.write("<p>こ");
document.write("れ");
document.write("は<br>");
document.write("順");
document.write("次");
document.write("の</p>");
document.write("<p>例");
document.write("で");
document.write("す</p>");
</script>
</body>
</html>
```

**図 44　順次のプログラム例**

JavaScript は HTML の中に埋め込みます。埋め込むためには script タグを使います。JavaScript 部分を HTML から分離してファイルにして、HTML でそのファイルを読み込む方法もあります。document.write()は文字を出力する JavaScript の命令です。HTML のタグを出力すれば、それがそのままブラウザに解釈されて出力されます。図 45 に実行結果を示します。ちゃんと文章になっているのは、命令が上から下へ実行されているからです。下から上へ実行されているのだったら、「すで例の次順...」と表示されているはずですね。

これは
順次の

例です

**図 45　実行結果**

p タグと br タグもちゃんと機能しています。p タグによって段落が 2 つできます。1 つ目の段落の中で、「これは」の次で強制改行されています。

## 12.2. 変数

変数は値を入れる箱のようなものです。図 44 のプログラムは document.write()で直接 HTML を書きましたが、ここでは変数に HTML 文字列を入れてみましょう。図 44 を変数を使った書き方に変えたプログラムが図 46（js2.html）です。

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>変数</title>
</head>
<body>
<script>
let mojiretsu;
mojiretsu = "<p>こ";
mojiretsu += "れ";
mojiretsu += "は<br>";
mojiretsu += "順";
mojiretsu += "次";
mojiretsu += "の</p>";
mojiretsu += "<p>例";
mojiretsu += "で";
mojiretsu += "す</p>";
document.write(mojiretsu);
</script>
</body>
</html>
```

**図 46　変数を使うようにしたプログラム**

「mojiretsu」が変数です。「=」は代入を表します（難しい言い方で代入演算子）。最初、mojiretsu に" こ"を代入しています。次から続く「+=」も演算子で、文字列を対象とする場合は文字列の連結（くっつけてつなげること）を行います。これも一種の代入です。" こ"に"れ"を連結すると" これ"になります。最後の"す "の連結までを行うと、変数 mojiretsu の中身は" これは<br>順次の 例です "になります。このプログラムの実行結果は、図 45 と同じになります。

### 12.3. その他のルール

JavaScript は大文字小文字を区別します。図 47 のプログラム（js3.html。HTML 部分は省略しています）を実行すると、図 48 の出力を得ます。大文字の DOCUMENT.WRITE() という命令はないので、エラーとなります。

```
let a;
let A;
a = "a です<br>";
A = "A です<br>";
document.write(a);
document.write(A);
DOCUMENT.WRITE(a); /* この行はエラーになります */
```

図 47　大文字小文字は識別される

aです
Aです

図 48　出力結果

エラーになっているかは、ブラウザで F12 キーを押してコンソールを表示させるとわかります（図 49）。


図 49　ブラウザのコンソールに表示されるエラー

JavaScript の仕様として、文末にセミコロン「;」は打たなくてもいいです。しかし、プログラミングスタイルとして文末にセミコロンを打つことを推奨する人は多いです。本テキストでも、文末にセミコロンを使います。

/* */に囲まれた部分はコメントとなり、実行されません。// という書き方もあり、この場合は//より右（改行が来るまで）はコメントとなります。コメントはコンピュータに対しては意味を持ちませんが、人間に対しては意味があります。なぜ命令をそう書いたかやプログラムでやっていることの説明をします。人間は忘れる生き物なので、例え自分が書いたプログラムでも何日もたてばなぜそう書いたか忘れることがあります。そのため、適切なコメントを書くことは大切です。

コメントの例です：

```
/* これはコメントです。 */
/* 見る人、書く人のためのメモです。 */
/* コメントはコンピュータは解釈しません。 */
let pi;
pi = 3.14;    // 円周率。これもコメントです。
document.write(pi);
```

[考えてみましょう]

こんなコメントはだめです。(comment.html)
プログラムの処理内容とコメントの内容が合っていないからです。どんなコメントが適切でしょうか?

```
/* 円の面積を求める */
let pi = 3.14
let chokkei = 5
let enshu = chokkei * pi
document.write(enshu)
```

コメントと処理が合っていないと、読む人が混乱します。

## 13. 比較（分岐）

### 13.1. 比較の概要

2 つ目の基本形は、比較です。条件の比較によって、処理が分岐(ぶんき)します。図 50 は比較（分岐）のフローチャートです。この例では、お金があればポテチを買えますが、お金がなければポテチは買えません。はいといいえで処理が分岐していることを確認してください。

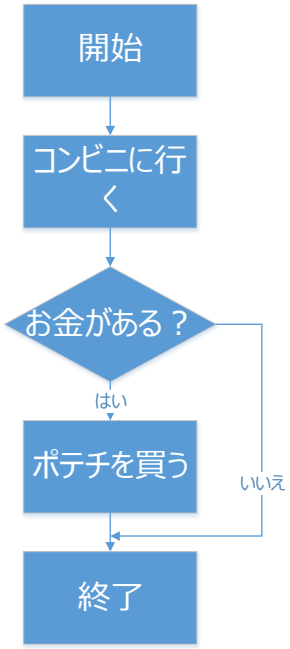



**図 50　比較（分岐）のフローチャート**

比較の JavaScript プログラムの例を図 51 に示します。(js4.html)

```
let potachi = 120;
let okodukai = 110;
let message;
if (okodukai < potachi){
    message = `おこづかい(${okodukai}円)よりポテチ(${potachi}円)の方が高いので、ポテチは買えません。`;
}else{
    message = `おこづかい(${okodukai}円)でポテチ(${potachi}円)が買えます。`;
}
document.write(message);
```

おこづかいの額とポテチの価格の比較

**図 51　比較のプログラム例**

変数 potachi がポテチの価格を表し、変数 okodukai が持っているおこづかいの額を表しています。ポテチは 120 円、おこづかいは 110 円しか持っていないため、okodukai < potachi の条件比較は「真」となり、買えない旨のメッセージが表示されます（図 52）。つまり、条件によって処理が分岐することになります。




**図 52　実行結果**

バッククォート「`」とダブルクォート「"」の違いは、文字列の中で変数を展開できることです。

[練習]

- 変数 okodukai に代入する値を 120 に変更し、図 52 とは結果が異なることを確認してください。結果が異なるのは、条件によって処理が分岐しているからです。
- バッククォートをダブルクォートに変更してみてください。

[真と偽について]

条件が正しいことを「真」、正しくないことを「偽」といいます。変数 a と b が等しいかチェックするのは

```
if (a == b)
```

と書きます。a と b が等しいと、a == b は真となります。a と b が等しくないと、a == b は偽となります。if については次節で説明します。

### 13.2. 比較のパターン

if 文の書き方は、まず一番単純なものが図 53 です。条件が真なら、命令 1 が実行されます。条件が偽なら何も実行されません。波括弧{}の中は処理ブロックと呼ばれ、処理ブロックの中は命令がいくつあってもかまいません。

```
if (条件){
    命令 1
}
```

**図 53　if のみ**

図 54 は if～else のパターンです。条件が真なら命令 1 が実行され、条件が偽なら命令 2 が実行されます。

```
if (条件){
    命令 1
}else{
    命令 2
}
```

**図 54　if～else**

図 55 は if～else if～else のパターンです。条件 1 が真なら命令 1 が実行され、そうでないなら条件 2 のチェックに移ります。条件 2 が真なら命令 2 が実行され、そうでなくどの条件にも合致しない（条件 1 も条件 2 も偽）なら命令 3 が実行されます。このパターンでは、else if は複数書けるので、条件を増やせます。

```
if (条件 1){
    命令 1
}else if (条件 2){
    命令 2
}else{
    命令 3
}
```

**図 55　if～else if～else**

### 13.3. switch～case による条件分岐

if 以外の条件分岐の命令に switch と case というものがあります。if との違いは、1 つの条件で多数の分岐を作れることです。メリットは、if 文で else if を増やすよりコードが見やすくなります。

図 56 は switch～case を使ったプログラムの例です（js5.html）。乱数を使っており、実行するたびに出力が変化します（図 57）。

```
let r = Math.floor(Math.random() * 4) + 1; // 1～4 の乱数を生成する。
let message;
switch (r){
case 1: message = "1 ですな";
        break;
case 2: message = "2 でござる";
        break;
case 3: message = "3 ある";
        break;
case 4: message = "4 っすね";
        break;
}
document.write(message);
```

**図 56　switch～case**




**図 57　実行結果**

[練習]

- Math.floor()と Math.random()の働きを調べてください。
- break がなかったらどうなりますか。コメントアウト[10]してみましょう。

[10] コメントアウトは、命令をコメントにして実行させなくすることです。

## 14. 繰り返し

### 14.1. 繰り返しの概要

3 つ目の基本形は、繰り返しです。繰り返しのフローチャートは、比較を使って描けます（図 58）。ポテチを買うお金があるうちはポテチを買い続け、残金がポテチの価格を下回ったら購入を終了します。

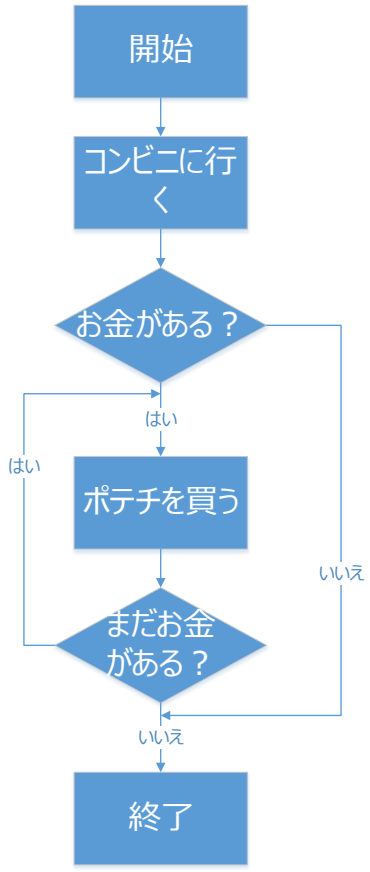



**図 58　繰り返しのフローチャート**

繰り返しの JavaScript プログラムの例を図 59 に示します。(js6.html)

```
let potachi = 120;
let okodukai = 500;
let message;
while (okodukai >= potachi){
    message = `おこづかい(${okodukai}円)でポテチ(${potachi}円)が買えます。<br>`;
    document.write(message);
    okodukai -= potachi; // おこづかいからポテチ料金を引きます。
}
message = `おこづかい残金(${okodukai}円)がポテチの価格(${potachi}円)を下回りました。
<br>`;
document.write(message);
```

繰り返す条件

**図 59　繰り返しのプログラム例**

このプログラムの実行結果を図 60 に示します。最初、おこづかいを 500 円持っています。ポテチは 120 円です。ポテチを買えるので買います。おこづかいは残り 500－120=380 円になります。380 円でまだポテチを買えるので買います（その繰り返し）。おこづかいが最終的に 20 円になり、買えなくなります。okodukai >= potachi という条件を満たさなくなるので、while ループから抜けます。

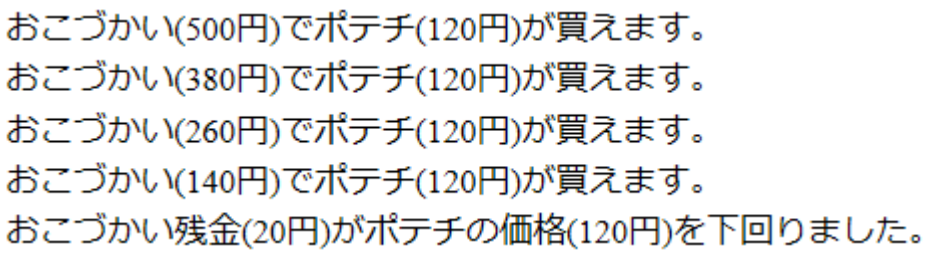



**図 60　実行結果**

### 14.2. 繰り返しのパターン

繰り返しには、以下のパターンがあります。

- 条件前判断（while）
- 条件後判断（do～while）
- 回数指定（for）

条件前判断（while）の書き方を図 61 に示します。条件が真の間、命令が実行され続けます。条件が偽になったとき繰り返しは終了します。最初から条件が偽のとき、命令は実行されません。

```
while（条件）{
    命令
}
```

**図 61　条件前判断（while）**

条件後判断（do～while）の書き方を図 62 に示します。条件が真の間、命令が実行され続けます。条件が偽になったとき繰り返しは終了します。最初から条件が偽であったとしても、命令は 1 回実行されます。

```
do{
    命令
}while（条件）;
```

**図 62　条件後判断（do～while）**

回数指定の書き方を図 63 に示します。まず、ループカウンタの初期化が実行されます。次に条件が真の間、ループカウンタの増減と命令が実行されます。

```
for（ループカウンタの初期化; 条件（回数指定）; ループカウンタの増減){
    命令
}
```

**図 63　回数指定**

回数指定のプログラム作成例を図 64 に示します。1 個 120 円のポテチを 5 個買ったらいくらになるかを for を使って計算しています。(js7.html)

```
let potachi = 120;
let okodukai = 0;
let message;
let counter;
for (counter = 1; counter <= 5; counter++){
    okodukai += potachi;
}
message = `1 個${potachi}円のポテチを${counter - 1}個買うと、おこづかいは${okodukai}
円必要です。<br>`;
document.write(message);
```

**図 64　for ループでポテチ 5 個分のお金を計算**

[練習]

- 図 64 のプログラムを実行して、結果を確認してください。
- 図 64 のプログラムで、メッセージを表示するとき変数 counter から 1 を引いています。なぜ counter でなく counter - 1 を表示しているのでしょうか。

# 15. 抽象化

## 15.1. 抽象化の概要

4 つ目の基本形は、抽象化です。抽象化とは、一連の処理のまとまりに名前をつけることです。親が子供に手伝いを頼むときのことを例にしましょう。子供にお金を渡し、「商品 A、商品 B、...をこのお金で買ってきて。もし万一お金が足りなかったら、商品 A を買わなくてもいい。商品 B は必ず買ってくるんだよ。レシートをもらってきてね。お釣りをもらうのも忘れないでね。」などと指示します。これを毎回説明するのは大変なので、簡単な買い物を経験させ、おつかいとは何か説明したあとで「おつかい行ってきて。買うものはこのメモに書いてあるから」と頼みます。「おつかい」が、この一連の行為につけた名前です。名前をつければ、次からその名前を呼ぶだけですむので便利です。

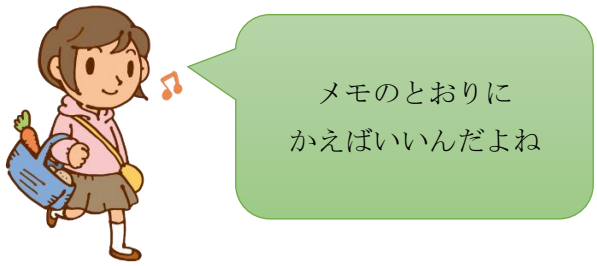



**図 65 「おつかい」の意味がわかった子供**

JavaScript での抽象化の手法の一つに関数があります。おつかいを例にとってみましょう。話を簡単にするために、買う商品は一種類、所持金で買える限りの商品を買うプログラムを考えます。一連の処理に otsukai という名前をつけます。作成例を図 66 に示します。このプログラム (js8.html) は 38 ページの図 59 を少し変えただけです。実行結果も 39 ページの図 60 と同じです。

```
let message;
message = otsukai(500, "ポテチ", 120);
document.write(message);

function otsukai(shojikin, shouhin, kakaku){
    let message = "";
    while (shojikin >= kakaku){
        message += `所持金(${shojikin}円)で${shouhin}(${kakaku}円)が買えます。<br>`;
        shojikin -= kakaku;
    }
    message += `所持金(${shojikin}円)が${shouhin}の価格(${kakaku}円)を下回りました。
<br>`;
    return message;
}
```

500 円で 120 円のポテチを買えるだけ買ってきて

点線の中が関数 otsukai の実体。

**図 66 おつかいのプログラム**

2 行目で関数 otsukai()を呼び出しています。関数 otsukai()の実体（定義）は 5 行目以降です。関数 otsukai()は 3 つの引数を持ちます。1 つ目は所持金、2 つ目は買う商品名、3 つ目は買う商品の価格です。2 行目で所持金 500 円、買う商品名"ポテチ"、価格 120 円を渡しています。関数 otsukai()は、それらを変数 shojikin、shouhin、kakaku にそれぞれ受け取ります。while 文で買えるだけ買い、変数 message に HTML メッセージを作ります。最後に return message としてメッセージを呼び出し元に返却（へんきゃく）します。呼び出し元では、関数 otsukai()から返却されたメッセージを受け取り、それをドキュメントに出力します。

[小テスト 3]

図 66 の 3 行目で、ドキュメントを出力した後、処理はどこへ流れるでしょうか。

## 15.2. フォームと JavaScript の連携

関数がわかったので、だんだん本格的なプログラム作成に入っていきます。ユーザから 2 つの値を入力してもらい、その和を求める「足し算プログラム」を作ってみましょう。form タグを使い、画面上に 4 つの部品を配置します（図 67）。図 68 はこのプログラムで使う部品を表にまとめたものです。「+」は部品とせず、HTML ドキュメントとします。


**図 67　足し算プログラムの画面**

| 部品 | id | 初期表示 | 用途 |
|---|---|---|---|
| text | v1 | なし | 1 つ目の数値を入力する |
| text | v2 | なし | 2 つ目の数値を入力する |
| button | なし | = | 和を求めるイベントを発生させる |
| text | answer | なし | 和を表示する |

**図 68　部品一覧表**

足し算プログラムのアルゴリズム（やること）を考えてみましょう。

1. テキスト v1 から値を取得し、変数 v1 に入れる。
2. テキスト v2 から値を取得し、変数 v2 に入れる。
3. 変数 v1+変数 v2 の計算結果をテキスト answer に入れて表示する。

たったこれだけですね。ではプログラム（図 69）を見てください（js9.html）。

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>足し算プログラム</title>
<script>
function tashizan()
{
    let answer;
    let v1 = document.getElementById("v1").value;
    let v2 = document.getElementById("v2").value;
    answer = Number(v1) + Number(v2);
    document.getElementById("answer").value = answer;

    return;
}
</script>
</head>
<body>

<form>
 <p>
 <input type="text" id="v1" size=4 style="text-align: right;"> +
 <input type="text" id="v2" size=4 style="text-align: right;">
 <input type="button" value="=" onClick="tashizan();">
 <input type="text" id="answer" size=5 style="text-align: right;">
 </p>
</form>

</body>
</html>
```

**図 69　足し算プログラム**

「=」ボタンを押したときは「Click」イベントが発生します。「=」ボタンのパラメータに onClick="JavaScript プログラム..."としておくと、「=」ボタンを押したときに onClick に書いた JavaScript プログラムが動きます。これをイベントハンドラといいます。しかし、イベントハンドラに書いたプログラムが長いと読みづらくなります。そこで、足し算処理は抽象化して関数 tashizan()を呼び出すことにします。

関数 tashizan()の中を説明します。document.getElementById().value を使うと、form 内の画面項目から値を取得したり値をセットすることができます。v1 と v2 を足そうとすると、そのままでは文字列の連結になります。そこで、文字列から数値に変換するために Number()関数を使っています。関数 tashizan()は、呼び出し元へ何も返却しません。

足し算プログラムは理解できたでしょうか。フォームと JavaScript の連携はとても重要で、応用が利きます。ですので、別のプログラムでもう一度確認しましょう。HTML の部（9 章のフォーム）で説明した html7.html に JavaScript のプログラムを追加した js10.html を見てください。図 70、図 71 は js10.html から JavaScript 部分だけを抜き出したものです。その他の部分は html7.html と同じです。

```
// 各項目値の取得
function GetItems()
{
    let str = "";

    // 氏名
    str = "氏名：" + document.getElementById("氏名").value + "¥r¥n";

    // 性別
    let sei;
    sei = document.getElementsByName("性別");
    str = str + "性別："
    if (sei[0].checked){
        str = str + "男¥r¥n";
    }else{
        str = str + "女¥r¥n";
    }

    // 所持品
    let properties;
    properties = document.getElementsByName("所持品");
    str = str + "所持品："
    if (properties[0].checked){
        str = str + properties[0].value
    }
    if (properties[1].checked){
        str = str + properties[1].value
    }
    if (properties[2].checked){
        str = str + properties[2].value
    }

    // 好きな果物
    str = str + "好きな果物：" + document.getElementById("好きな果物").value +
"¥r¥n";

    // 自由意見
    str = str + "自由意見：" + document.getElementById("自由意見").value;

    alert(str);
    return;
}
```

変数 str に出力文字列がたまっていく

最後に alert()で str を表示

図 70　各画面項目の値を取得して表示する JavaScript プログラム

```
<input type="button" value="送信" onClick="GetItems();">
```

**図 71　送信ボタンを押すと図 70 の GetItems()関数を呼ぶ**

js10.html の説明をします。このプログラムは長くて複雑そうに見えますが、画面項目が氏名、性別、所持品、好きな果物、自由意見、送信ボタン、取り消しボタンの 7 つがあるためであり、1 つ 1 つの処理は意外と簡単なものとなっています。9 章のフォームの所の復習にもなりますので、重複した説明があります。

HTML の中で、JavaScript はどこに書いてもいいことになっています。関数は head タグの中にまとめて書くことが多いです。script タグの開始終了で囲んだ部分に書きます。

```
<head>
<title>〇〇〇</title>
<script>
ここに JavaScript を書く。
</script>
</head>
```

送信ボタンを押したとき、onClick により GetItems()という関数が動くようになっています。GetItems()は各画面項目の値を取得し、alert()命令によって項目を表示します。各項目の値は変数 str に集まります。str = str + 〇〇; とすると、str に文字を連結できます。str += 〇〇; と書いても同じです。

```
// 各項目値の取得
function GetItems()
{
    .... (画面項目の取得)
    alert(str);
    return;
}
```

氏名は HTML 側は単純入力項目で「氏名」という id を持っています。document.getElementById("氏名").value で「氏名」という id を持つ項目の値を得ることができます。"¥r¥n"は復帰・改行コードです。

```
氏名：<input type="text" id="氏名">

str = "氏名：" + document.getElementById("氏名").value + "¥r¥n";
```

画面項目は、同じ id を持つことはできません。同じ id を持つ項目があった場合、HTML としてエラーになるわけではありませんが JavaScript でその id を使用しようとするとうまく動きません。

性別は男と女のラジオボタンです。ラジオボタンとして動かすためには、全部のボタンが同じ name を持つ必要があります。ラジオボタンは「性別」という name を持っています。ここでは、getElementById()ではなく getElementsByName()を使っています。getElementsByName()を使うと、男のボタン状態、女のボタン状態を配列で取得できます。変数 sei に配列を取得するので、sei[0].checked が男ボタンの押された状態、sei[1].checked が女ボタンの押された状態となります。ラジオボタンの動きにより、ボタンはどれか 1 つが押された状態になることが保証されます。もし sei[0].checked が真なら男が押されているし、そうでないなら女が押されていると判断できます。

```
<input type="radio" name="性別" value="男" checked="checked"> 男
<input type="radio" name="性別" value="女"> 女

    sei = document.getElementsByName("性別");
    str = str + "性別："
    if (sei[0].checked){
        str = str + "男¥r¥n";
    }else{
        str = str + "女¥r¥n";
    }
```

所持品も性別と同様です。所持品という name を持つチェックボックスです。getElementsByName()で各ボタンの状態を配列に取得します。所持品はラジオボタンではないので、複数選ぶこともできますし 1 つも選ばないこともできます。ですので、全部のボタンに対して押されているかを調べる必要があります。

好きな果物はドロップダウンリストであり、選択肢が複数あっても 1 つの項目として扱えます。そのため、「好きな果物」という id を持たせられ、getElementById()で値をそのまま取得できます。

```
<select id="好きな果物">
   <option>りんご</option>
   <option>みかん</option>
   <option>バナナ</option>
   <option>パイナップル</option>
</select>

str = str + "好きな果物：" + document.getElementById("好きな果物").value + "¥r¥n";
```

自由意見は複数行に対応した入力項目です。単純入力項目と同じように扱えます。

送信ボタンは input に type="button"と指定することで作成できます。onClick は押したらどうするという JavaScript を書きます。取り消しボタンは type="reset"とすると、押したときに各画面項目を初期化できます。

[小テスト 4]

js10.html の JavaScript にはバグがあります。バグというのは、プログラムのミスのことです。どんなバグですか。

## 16. ライブラリを使おう

JavaScript は世の中に登場して 20 年以上経っており、いろいろな人がいろいろなプログラムをたくさん開発しています。中には、JavaScript のプログラミングが容易になったり便利になったりするプログラムを作ってくれた人もいます。そのような、プログラミングが容易になったり便利になったりするプログラムをライブラリといいます。有料のものと無料のものがありますが、ここでは無料で使えるものを紹介します。

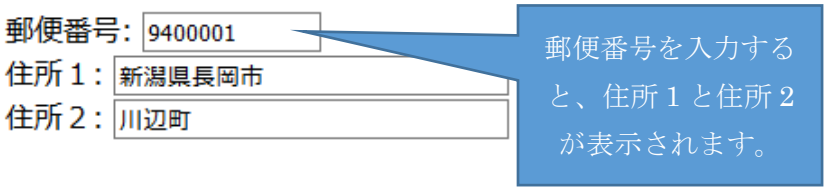
### 16.1. 住所補完 JavaScript

住所補完 JavaScript の URL= https://js-doc.postcode-jp.com/#javascript


図 72　住所補完 JavaScript のホームページ

Web サイトで、ユーザの住所を入力させたいことが良くあります。例えば、アマゾンや楽天などの買い物ができるサイトで、商品の送り先を入力させる場合です。ところが、ユーザは現金なもので、入力が面倒だと買い物をやめてしまうことがあります。これをサイトからの離脱（りだつ）といいます。買ってもらう側からすれば、離脱されるのは困るので、どうにかして住所入力を簡単にしたいのです。そこで、簡単に住所を入力できるライブラリがあると便利です。

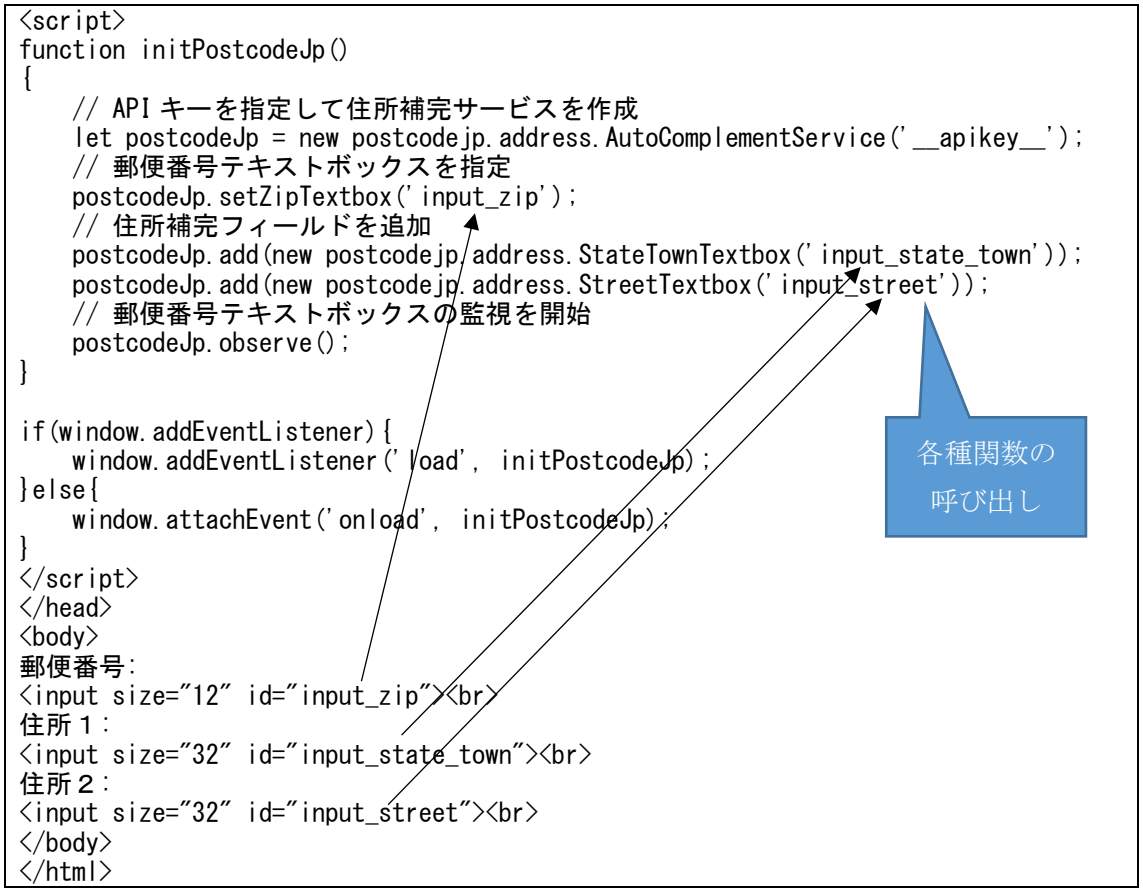
住所補完 JavaScript は、郵便番号を入力すると県・市と町名を入力項目に表示してくれます。ユーザは番地と建物名・部屋番号を入力するだけですむので楽です。




住所補完 JavaScript を使うサンプルソースを図 73 に示します。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>住所補完 JavaScript</title>
<!-- 住所補完 JavaScript ライブラリの読み込み -->
<script src="https://postcode-jp.com/js/postcodejp.js"> </script>
```

ライブラリの読み込み




```
<script>
function initPostcodeJp()
{
    // API キーを指定して住所補完サービスを作成
    let postcodeJp = new postcodejp.address.AutoComplementService('__apikey__');
    // 郵便番号テキストボックスを指定
    postcodeJp.setZipTextbox('input_zip');
    // 住所補完フィールドを追加
    postcodeJp.add(new postcodejp.address.StateTownTextbox('input_state_town'));
    postcodeJp.add(new postcodejp.address.StreetTextbox('input_street'));
    // 郵便番号テキストボックスの監視を開始
    postcodeJp.observe();
}

if(window.addEventListener){
    window.addEventListener('load', initPostcodeJp);
}else{
    window.attachEvent('onload', initPostcodeJp);
}
</script>
</head>
<body>
郵便番号:
<input size="12" id="input_zip"><br>
住所１:
<input size="32" id="input_state_town"><br>
住所２:
<input size="32" id="input_street"><br>
</body>
</html>
```

図 73　住所補完 JavaScript を使うサンプルソース(lib1juusho.html)

関数 initPostcodeJp()の中身を詳しく知る必要はないです。ライブラリを使うと、難しいことが簡単になるとわかってもらえばいいです。意外にソースが短いことにおどろいたのではないでしょうか？6 行目で<script src="https://postcode-jp.com/js/postcodejp.js"></script>としているところが、住所補完 JavaScript ライブラリを読み込んでいるところです。ネットに接続していることが動作条件です。入力項目・郵便番号（input_zip）を監視して、入力されたら郵便番号に合う県・市（input_state_town）と町名（input_street）を表示しています。

## 16.2. Plotly

Plotly の URL= https://plotly.com/graphing-libraries/

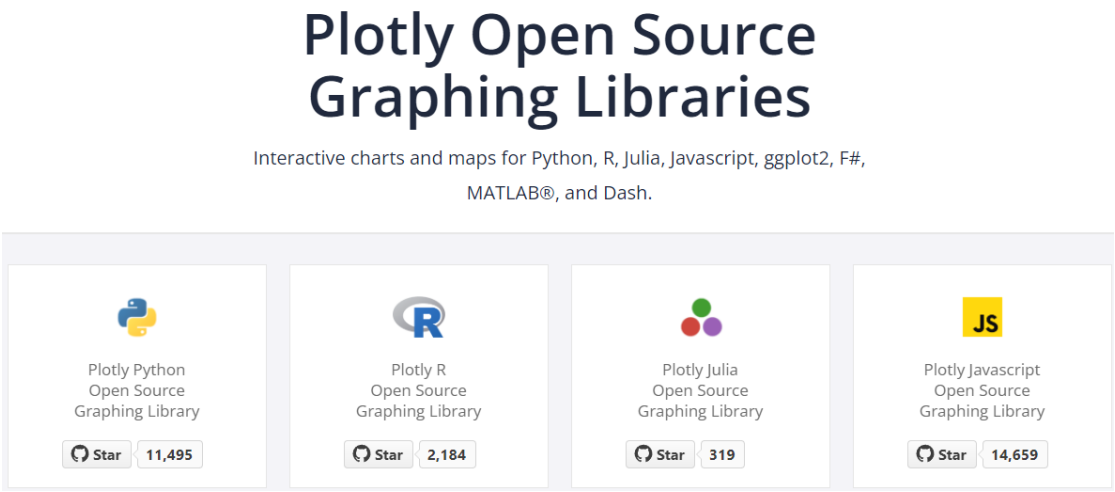



**図 74　Plotly のホームページ**

Plotly はグラフを描けるライブラリです。グラフを描くライブラリは他にもあり、世界的に有名なものに Chart.js があります。Chart.js は線グラフ、棒グラフ、円グラフなどさまざまなグラフを簡単に描けます。しかし、Plotly は Chart.js より数学との親和性(しんわせい)が高いので、このテキストでは Plotly を採用しました。Chart.js の使用サンプルソースを lib4chart.html として用意しておきましたので、開いてみてください。10 段階評価の通知表の成績を比較する棒グラフを図 75 のように描くものです。去年が赤、今年が水色です。

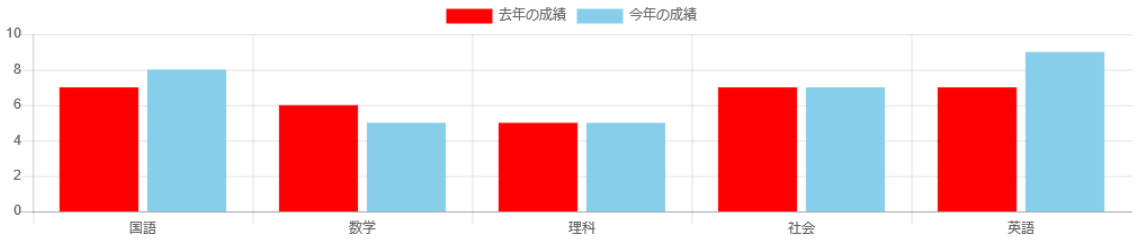



**図 75　Chart.js の例・成績比較**

Plotly で一次関数のグラフを描いてみましょう。

---

*数学の関数とは、x と y があるときに、x の値が決まると y の値がただ 1 つ決まることを「y は x の関数である」といいます。*

---

一次関数の式は「y=ax+b」で表せます。a は傾(かたむ)き、b は切片(せっぺん)です。a=1、b=0 だと y=x という一番シンプルな式になりますね。

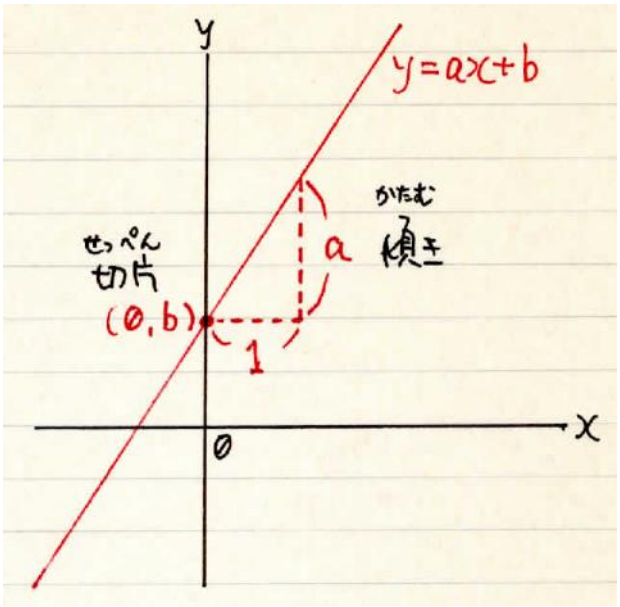



**図 76　y=ax+b のグラフ**

例として、図 78 の y=3x-2 のグラフを描くソース（図 77）を lib2plotly.html として用意しています。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>y=3x-2 のグラフ</title>
<script src="https://unpkg.com/mathjs@10.4.3/lib/browser/math.js"></script>
<script src="https://cdn.plot.ly/plotly-1.35.2.min.js"></script>
<script>
function draw() {
    // 式を検査します。
    let expression = "y=3x-2";
    let expr = math.compile(expression);

    // x の範囲は-10 から 10 まで 0.5 きざみとします。
    let xValues = math.range(-10, 10, 0.5).toArray();
    // y=3x-2 の関数から、y の値を求めます。
    let yValues = xValues.map(function (x) {
        return expr.evaluate({x: x});
    });

    // Plotly を使って図を描きます。
    let _data = {
        x: xValues,
        y: yValues,
        type: "scatter"
    };
    let data = [_data];
    Plotly.newPlot("plot_area", data);
}
</script>
</head>
<body onLoad="draw()">

<center><h1>y=3x-2 のグラフ</h1></center>
<div id="plot_area"></div>

</body>
</html>
```

式 y=3x-2 を定義する

plot_area に data を使って
グラフを描く

HTML が読み込まれたとき
に関数 draw()を実行する

グラフを描く場所
「plot_area」を定義する

図 77　グラフを描くサンプルソース(lib2plotly.html)

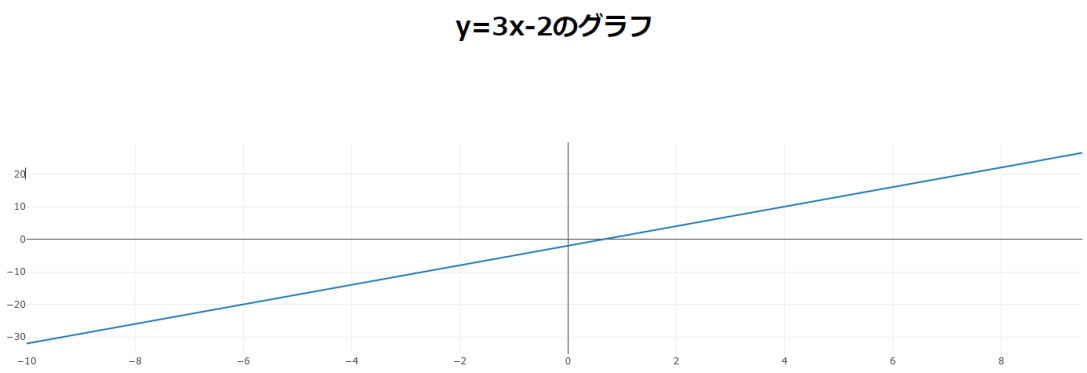



図 78　y=3x-2 のグラフ

BODY タグに onLoad="関数名"と書くと、HTML が読み込まれたときに関数を実行することができます。グラフを描くためのエリアとして DIV タグで plot_area を定義しています。DIV タグはここで初めて出てきましたが、ある段落に名前をつけるときに使用します。"y=3x-2"という式を定義して、x がどのような値を持つかを設定し、その x によって y の値を求め、グラフを描いています。

## 16.3. カレンダー

| 日 | 月 | 火 | 水 | 木 | 金 | 土 |
|---|---|---|---|---|---|---|
| | | | | | 1 | 2 |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| 31 | | | | | | |

図 79　2022 年 7 月のカレンダー

ここで紹介するカレンダーライブラリは、有名なものではなく一般の方が作ったものです。cal.js が JavaScript だけをファイルにしたもので、lib3calendar.html が cal.js を読み込んでライブラリを呼び出します。図 80 にサンプルソース（一部）を示します。

```
<head>
<title>カレンダー</title>
<script src="./cal.js"></script>
</head>
<body>
<div id="calendar"></div>
<script>
// 年月の指定
let year = 2022;
let month = 7;

window.onload = function() {
    let data = generate_month_calendar(year, month);
    document.getElementById("calendar").appendChild(data);
}
</script>
</body>
```

関数呼び出し

図 80　カレンダーを描くサンプルソース(lib3calendar.html)

表示したいカレンダーの年と月を数値で関数 generate_month_calendar()に渡すと、カレンダーを書く HTML を返してくれます。window.onload...は BODY タグに onLoad を書くのと同じ意味です。こちらの方は、関数名をつける必要がありません。

## 17 小テストの答え

1. table タグに border="数"という属性を付与します。数は枠線の太さです。

```
<table align="center" border="1">
```

2. a タグの中に img タグを入れます。

```
<a href="https://www.google.co.jp/">
<img src="https://www.google.co.jp/images/branding/googlelogo/1x/googlelogo_color_272x92dp.png">
</a>
```

3. document.write(message)の次にはもう処理がありません。関数 otsukai()の定義がありますが、そこへは処理が流れません。関数というのは、呼ばないかぎり実行されないのです。つまり、3 行目の document.write(message)でプログラムは終了です。それを確かめるため、js8-a.html を用意しました。

```
let message;
message = otsukai(500, "ポテチ", 120);
/* document.write(message); */

function otsukai(shojikin, shouhin, kakaku){
    let message = "";
    while (shojikin >= kakaku){
        message += `所持金(${shojikin}円)で${shouhin}(${kakaku}円)が買えます。<br>`;
        shojikin -= kakaku;
    }
    message += `所持金(${shojikin}円)が${shouhin}の価格(${kakaku}円)を下回りました。
<br>`;
    return message;
}

document.write(message);
```

コメントアウト

追加

図 81 関数 otsukai()の後ろに処理を追加したソース(js8-a.html)

プログラム js8-a.html を実行しても、js8.html と同じ結果になります。基本形「順次」の法則により、処理は上から下へ（関数定義を素通りして）流れるからです。

4. js10.html を実行してみるとすぐわかります。図 82 は実行結果（入力して送信ボタンを押したところ）です。

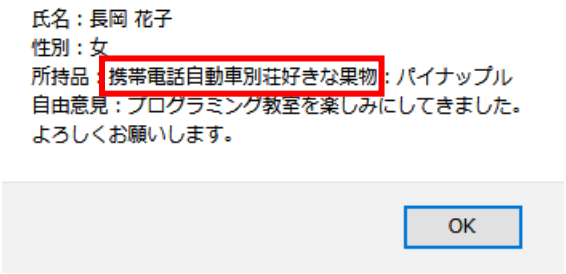



**図 82　js10.html の実行結果**

所持品の携帯電話、自動車、別荘がそのままつながっていますし、好きな果物の前に区切りがありません。このバグを修正するには…？

## 18. 課題

これから皆さんに課題に取り組んでもらいます。どれか１つをえらんでプログラムを作ってください。難易度のめやすを★の数で示します。★が多くなるほど難しくなります。

| No. | 課題 | 難易度 |
|---|---|---|
| 1 | 何かを紹介する Web サイトを作ってください。 | ★ |
| 2 | 円の面積を求めるプログラムを作ってください。 | ★ |
| 3 | lib2plotly.html にエラー処理を入れてください。 | ★★ |
| 4 | js10.html をデバッグしてください。 | ★★ |
| 5 | カレンダーを改良してください。 | ★★★ |
| 6 | 他にどんなライブラリがあるか調査してください。 | ★★★★ |
| 7 | 電卓を作ってください。 | ★★★★★ |

課題 1

★何かを紹介する Web サイトを作ってください。

たとえば、

・自己紹介

・自分が通っている学校・勤めている会社の紹介

・所属しているクラブ、部、団体の活動の紹介

などです。

★★余力があったら、CSS を調べて見た目を良くしてください。人に興味を持ってもらうためには、なるべく見た目を良くすることが必要です。

課題 2

★円の面積を求めるプログラムを作ってください。

半径は画面から入力できるようにしてください。円周率は 3.14 とします。

★★余力があったら、（学校で習った）何か他の計算も画面でできるようにしてください。

課題 3

★★lib2plotly.html にエラー処理を入れてください。式が"y=aaa"のように間違っていると、グラフは何も描かれませんがなぜ何も描かれないかユーザにはわかりません。エラーが起こっていることを表示するようにしてください。

課題 4

★★js10.html をデバッグ[11]してください。小テスト 4 で解説したとおり、各所持品がくっついていますし、好きな果物の前に区切りがありません。

★★★余力があったら、何か項目を追加してみてください。見た目がさみしいので、見た目を良くしてください。

課題 5

カレンダー（lib3calendar.html）を改良してください。

★皆さんが使っているカレンダーと同じように、土曜日は青、日曜日は赤で表示してください。

★★見た目がさみしいので、見た目を良くしてください。

★★★余力があったら、変数 year と month に値を代入していますが、ここを改良し、いつ実行しても今年の今月が表示されるようにしてください。何年何月かも表示してください。

★★★★さらに余力があったら、矢印ボタン← →を用意し、←を押すと前月、→を押すと次月が表示されるようにしてください。

課題 6

★★★★他にどんなライブラリがあるか調査してください。そして、そのライブラリを使ったサンプルプログラムを作ってください。

課題 7

★★★★★電卓を作ってください。＋、－、×、÷が使えるようにしてください。2 つの値の計算結果を表示するようにしてください。

★★★★★★★★★★余力があったら、2＋3×4 など、3 つ以上の数の計算ができるようにしてください。

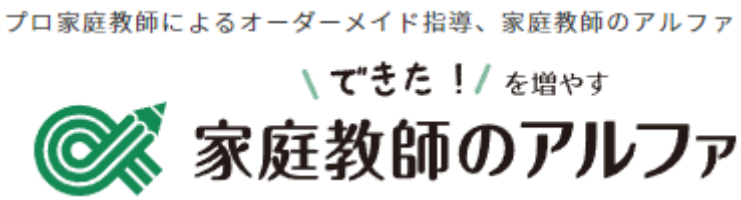
[11] デバッグ（debug）は、プログラムのバグをなくすことです。

[おことわりと謝辞]

おつかいのイラストは、イラストレインのものを使用しました。


https://illustrain.com/

一次関数 y=ax+b の説明は、家庭教師のアルファの説明を参考にしました。




https://alpha-katekyo.jp/

以上