Monacaで学ぶアプリ制作入門

～HTML×CSS×JavaScript編～

アシアル株式会社

Contents　目次

はじめに　スマホアプリ開発で楽しくプログラミング学習

スマートフォンやタブレットが登場してから10年以上が経ち、ほとんどの人がこれらを日常的に利用しています。スマートフォンやタブレットは素の状態でも通信機能とウェブサイトを閲覧するブラウザアプリを搭載していますが、「アプリ」と呼ばれるソフトウェアを追加インストールすることで、さまざまな機能やサービスを活用できるようになります。本書では、スマートフォンやタブレットで動作するモバイルアプリを自分で作る方法を解説します。

アプリを作るというとすごく難しそうな印象を持たれるかもしれませんが、必ずしも高度な知識や技術が必要になるわけではありません。作りたいアプリの用途や規模にもよりますが、画面数や機能が限られた簡単なアプリを開発したい場合なら数時間から数十時間の学習で実現が可能です。

本書ではスマホアプリ開発を通じてプログラミングの基礎を学べます。プログラミングを行うためのコンピューター言語として「HTML」「CSS」「JavaScript」という3 種類を活用します。それぞれの言語の役割は、以下のようになります。

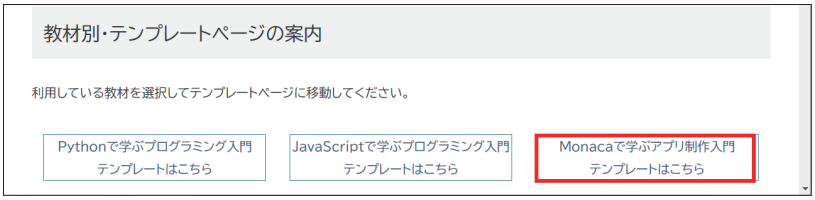
- HTML　文章や画像など、画面に表示する内容を定義します。
- CSS　画面に表示する内容の色・大きさ・配置といったスタイルを指定します。
- JavaScript　「ボタンをクリックしたときに結果を表示する」などのように、アプリに動きをつけます。

これらの言語はウェブサイトの制作にも利用できる、応用範囲の広い技術です。また特定の企業やソフトウェアに依存しない技術ですので、一度身に付ければ長く利用することが期待でき、最初に学ぶ言語としても最適です。

本書の活用　教材サポートページの紹介

教材サポートページにアクセスすることで、テキスト教材に従って学習する上で役に立つコンテンツを入手できます（https://edu.monaca.io/template/）。同ページにはMonaca Educationログイン後のダッシュボード右上にあるリンク集からもアクセスできます。

例として、各章で学ぶ実習のテンプレートとなるプロジェクトが掲載されています。



Monaca Educationにログインした状態で、テンプレートの「インポート」をクリックすると即座にプロジェクトを取り込めます。

教材サポートページには動画も掲載されています。なお、インポート用のページには、ダッシュボードの「インポート」ボタンからもアクセス可能です。

テキスト教材で学ぶ場合、インポートは利用頻度が高いため、こちらも活用して下さい。

第1章

アプリ開発入門

プログラミングを行うためには、まずプログラムを記述するためのソフトウェアが必要になります。そして記述したプログラムをコンピューターにインストールできる形に変換するソフトウェアや、動作確認を行うためのソフトウェアなども必要です。

こういったプログラミングに必要なソフトウェアを一つ一つ、自分のパソコンにインストールするのは大変です。そこで、プログラミングに必要なソフトウェアを統合的にまとめた「統合開発環境(IDE)」を使用します。

Monacaとは

Monacaはクラウドで動作する統合開発環境です。インターネット上で利用できるサービス全般を、雲の上にソフトウェアが置かれているイメージから、「クラウド」や「クラウドサービス」と呼びます。開発環境をクラウドに置くことで、自宅と学校のどちらからでもプログラミングを行うことが可能となっています。
Monacaは次のような特徴を備えています。

- パソコンに専用のソフトウェアをインストールする必要が無い。
- 少し古めのパソコンでも動作する。
- 開発中のプログラムを先生や友達と共有する機能がある。
- スマートフォンやタブレットで動くモバイルアプリが作れる。
- Webの標準的な技術でアプリ開発できる。

クラウドサービスだからどんなパソコンでも動いて共有も簡単

Monacaは「Google Chrome ブラウザ」というWebブラウザから利用できます。このWebブラウザがインストールされたパソコンであれば、OS（オペレーティングシステム）の種類やスペック（基本性能）は問いません。作ったアプリの動作確認は普段利用しているスマートフォンで行うことができます。
また、開発中のプログラムを他の人と共有する機能が搭載されています。作成途中のプログラムを先生や友達に見てもらいアドバイスをもらったり、エラーでつまずいてしまった時に助けてもらったりすることができます。アプリ開発のプロでも、自分以外の人にプログラムを見てもらうことで問題がすんなり解決することも多いものです。

モバイルアプリを標準的な技術で開発

スマートフォンで動くアプリを開発するには、さまざまな方法が存在します。AndroidやiOSといったスマートフォンのOSごとに別々のプログラミング言語を使わなければならない方法もありますが、MonacaではOSの種類を問わずに共通のプログラミング言語（HTML／CSS／JavaScript）を使ってアプリを開発します。Monacaで開発したアプリはAndroidとiOSのどちらでも動作するので、自分や友達、家族などがそれぞれ違う種類のスマートフォンを持っていても、同じようにアプリを動かすことができます。

Monaca の誕生と利用状況

Monacaはモバイルアプリの開発を便利にするために2011年に日本のアシアル株式会社という企業が開発したサービスです。現在、教育向けのMonaca Educationとあわせて40万人以上の人が利用しており、プロやセミプロだけでなく高校や大学・専門学校の授業でも幅広く使われています。

Monacaで作られたアプリも増え続けており、既に7万以上のアプリが世に出ています。最近では有名な企業のアプリでも使われており、代表的なアプリとしてテレビ朝日の映像・写真投稿サービス「みんながカメラマン」や、タニタの健康管理アプリ「ヘルスプラネット」、PayPay銀行の「残高確認アプリ」などが存在します。

Monaca Education

日本におけるプログラミング教育の必履修化にあわせて、2015年にスタートしたアシアルの教育事業です。当初は高校・大学・専門学校向けにMonacaの教材とライセンスを提供するだけでしたが、2019年にはMonaca Education専用サーバーを立ち上げ、教育向けに安価で高品質なクラウド型のプログラミング環境を日本中に提供しています。また、アシアル情報教育研究所を設立し、指導者に対する学習機会の提供やコミュニティの運営も行っています。

ぷよぷよプログラミング

2020年に日本の代表的なゲーム会社の一つである株式会社セガがリリースしたプログラミング教材です。この教材はMonaca Education上で動作する教材として無償で提供されており、アカウントがあればすぐに利用できます。興味があればぜひ挑戦してみてください。

Monacaではじめてのプログラミングを書こう

Monaca を利用するためにはアカウントが必要です。先生の指示に従ってアカウントを事前に準備してください。アカウント作成から進める場合は「次ページ」を参照してください。

Monaca Education にログインする

Monaca Education の公式サイトにアクセスします。

https://edu.monaca.io/

次に右上の「ログイン」をクリックして下さい。ログインフォームが表示されますので、事前に準備したアカウントの ID とパスワードを入力して「ログイン」します。

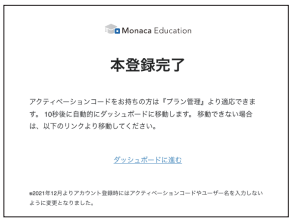
ログインに成功すると「ダッシュボード」が表示されます。

（参考）アカウント作成からはじめる場合

自身でMonaca Educationのアカウント作成を行うこともできます。ログインボタンの隣にある「アカウント作成」を選択するとアカウント作成フォームが表示されます。

授業の場合は先生の指示に従ってアカウントを作成します。メールアドレスで作成する場合はフォームを入力して「アカウント新規作成」を選択して下さい。「仮登録」状態となり本登録のためのメールが届きます。

メールアドレスで登録する場合は、受信メールに記載されたURLにアクセスすることで登録が完了となります。



※GoogleアカウントやMicrosoftアカウントで作成する場合はフォームを入力せずに作成ボタンを選択して、指示通りに登録を進めて下さい（仮登録はありません、また、メールも不要です）。

（参考）アカウント関連の設定

ダッシュボードではプロジェクトを一覧できます。また、右上のアイコンからアカウント関連の設定を変更できます。授業では先生の指示があった場合に利用してください。

「プラン管理」ではアクティベーションコードを利用したプラン変更が行えます。

自身のアカウントを後からGoogleアカウントなどと連携することも可能です。

プロジェクトの作成

まずはMonaca の使い方の学習も兼ねて簡単なプログラムを書いてみましょう。 最初に、プロジェクト（開発中のアプリのこと）を作成します。
[新しいプロジェクトを作る]ボタンをクリックして下さい。

プロジェクトのひな形となるテンプレートを選択する画面が現れます。

テンプレートには「サンプルアプリ」として完成した形になっているものから、開発のための土台のみを提供しているものまでさまざまな種類が用意されています。今回は「クラシック」を選択します。

今回はプロジェクト名を「はじめてのプログラム」に変更して「作成」ボタンをクリックします。プロジェクト名は自由につけることができます。あとで見た時にどんなプロジェクトか分かる名前や説明を書くようにしましょう。

プロジェクトが作成され、ダッシュボードに戻ります。

プログラムを記述する

「はじめてのプログラム」をクリックし［クラウド IDE で開く］ボタンをクリックします。

※［クラウド IDE で開く］には別モードも用意されています。iPad のソフトウェアキーボードに最適な「iPad モード」やプレビュー表示を行わない「セーフモード」などがあります。特に 12 章などで扱う「繰り返し処理」で無限ループが発生した場合はセーフモードが役に立ちます。

画面が切り替わり、Monaca クラウド IDE（以下、IDE）が表示されます。
IDE というのは「Integrated Development Environment」の略で日本語では統合開発環境と呼びます。IDE にはプログラミングに必要となるさまざまな機能が用意されています。

中央の一番大きいパネルが「エディタ」と呼ばれるプログラムを記述するためのパネルになります。

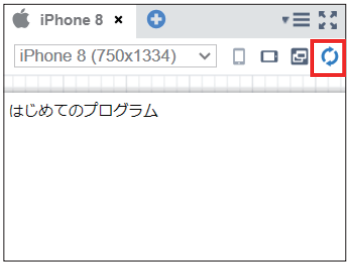
上部のパネルが「メニューバー」になります。プログラムを保存する機能やダウンロードする機能などはメニューバーから利用できます。特に授業でよく利用する機能は「授業」にまとまっています。

右側のパネルは「プレビュー」です。プログラムの実行結果が表示されます。「クラシック」テンプレートの場合、なにも表示されません。

<body> の部分に「はじめてのプログラム」と記述して、プレビューに表示してみましょう。メニューバーの左にある［保存ボタン］をクリックすると、エディタで編集した内容がクラウドに保存され、それによってプレビュー画面の再読込が行われ、表示内容が更新されます。

※　編集内容が保存されていない場合、コードエディタのファイル名の前に「*」マークが表示されます。

もし、プレビューが自動で更新されない場合はプレビュー画面右上にある青い円状の矢印ボタンをクリックしましょう。



小さな一歩ですが、コンピューターに対して指示を出すことができました。

JavaScript で簡単な命令を実行させる

先ほど変更したメッセージはHTMLで記述された文章です。HTML単体では「動き」のあるアプリを作ることができません。「動き」というのはアニメーションのことだけを指しているのではなく、ユーザーから入力された情報を受け取ったり、ユーザーの操作に合わせて文字や画像を後から差し替えたりすることなども「動き」といいます。動きのあるアプリは、HTMLにJavaScriptというプログラミング言語を組み合わせることで作成できます。
JavaScriptのプログラムはHTML文章の中にある<script>で囲まれた部分に記述します。

解説

index.html

```
10 <script>
11 </script>
```

alert()命令によるダイアログ表示

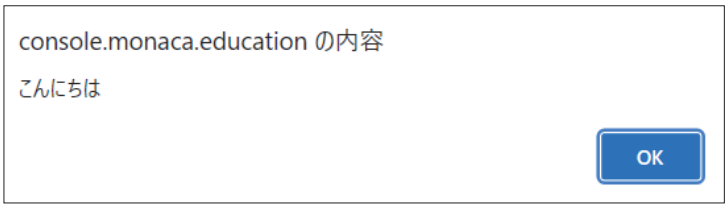
ダイアログとは、画面の前面に表示されるウィンドウのことです。ユーザーにメッセージを伝えたり、ユーザーからOKまたはキャンセルといった操作を促したりするために使われます。<script>と</script>の間に alert("こんにちは"); という記述を行って保存し、プレビュー画面で確認してみてください。

サンプルプログラム

index.html

```
10     <script>
11         alert("こんにちは");
12     </script>
```

実行結果

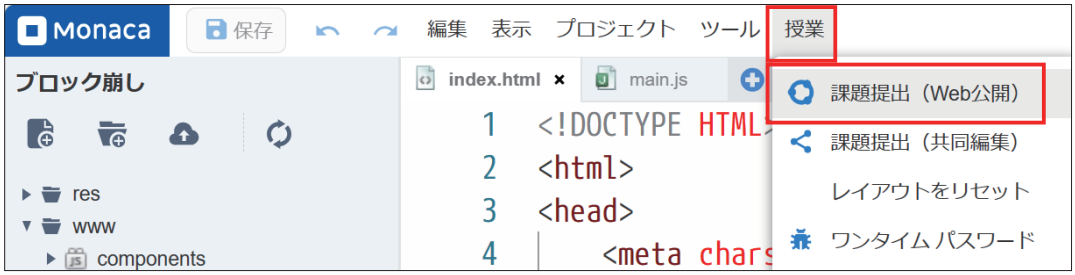


画面を開いたときに、ダイアログが表示されるようになりました。

作品を Web に公開する

作品を Web に公開することもできます。公開中の作品はログインなしで自身のスマートフォンなどから動作を確認できます。また作品の URL を先生に提出することもできます。

メニューの「授業」から「課題提出（Web 公開）」を選択して下さい。



公開のラジオボタンを「On」に切り替えて「適用する」を選択すると作品が公開されます。

二次元コードが表示されたら公開中です。

スマートフォンやタブレットで二次元コードを読み取れば作品を手元で動かすことができます。また、作品の URL を先生に提出するときには「URL をコピー」すると、端末のクリップボードにコピーされるので便利です

2 章以降の学習に向けて

2 章以降で利用するプロジェクトのテンプレートは、教材サポートページから入手可能です。こちらのテンプレートは画像素材が含まれていたり「答え合わせ機能」にも対応しており、実習をスムーズに進められます。教材サポートページは「インポート」などからアクセスできます。

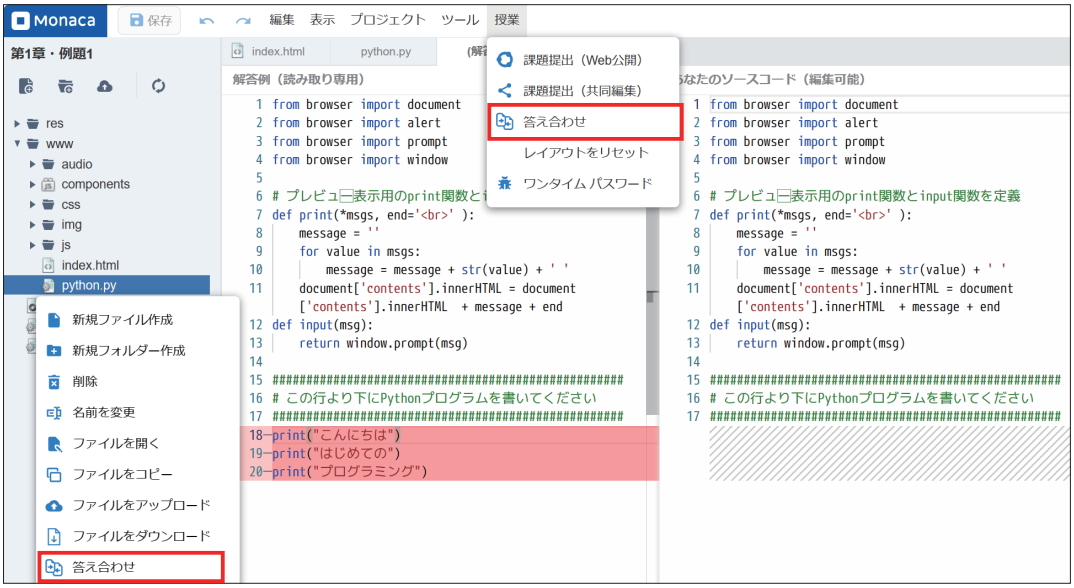
テキスト教材に従って進める場合は「Monaca で学ぶアプリ制作入門」を選択して下さい。また、「アプリ制作教材」では、おみくじアプリなどのサンプルアプリを提供しております。

テンプレート一覧から、学習したい内容のテンプレートを「インポート」します。

問題が無ければ「インポート」ボタンを押してプロジェクトをを作成して下さい。

答え合わせ機能

教材サポートページのプロジェクトは答え合わせ機能に対応しています。例えば答えを知りたいファイルのエディタタブにフォーカスを合わせてからメニュー「授業」の「答え合わせ」を選択すると左に解答例が表示されます。またはプロジェクトパネルのファイルを右クリックすることでも呼び出せます。



2

第 2 章

HTML 入門

Web ページやモバイルアプリの画面には、さまざまな色や画像が散りばめられ、とても華やかに装飾されていると思います。しかし実は、画面の元となっているファイル（「ソース」と呼びます）には文字だけでページの内容が記述されています。その記述言語が HTML と呼ばれるものです。

実習の前にサポートページから本章のために用意されたひな形をインポートしてください。

HTMLとは

HTML（Hyper Text Markup Language）はマークアップ言語の 1 つです。マークアップ言語では、文書が持つ内容をタグと呼ばれる特殊な文字列で囲む形式で記述します。
元々 HTML は、膨大な量の文書を閲覧しやすくする目的で開発されました。例えば、文書の中に専門用語が出てきた場合、その専門用語について解説されている別の文書をすぐに参照することが出来れば便利です。これを可能にしたのがHTML による「リンク」です。HTML にはリンク以外にも、文書を構造化したり、画像を参照したりする機能などがあります。

HTML の書き方

HTML では、文章やリンク、画像などの画面に表示する内容を「タグ」という文字列で囲みます。タグとは、画面に表示する内容の種類や役割を表す特殊な文字列です。
タグにはさまざまな種類がありますが、記述方法はどれも同じです。

文法　タグの記述方法と名称

```
<開始タグ>内容</終了タグ>
```

例　タグの記述例

```
<p>これは段落です。</p>
```

「開始タグ」と「終了タグ」の部分にはタグの名称が入ります。終了タグの前にはスラッシュを記述します。開始タグから終了タグまでの全体を「要素」と呼びます。
また、タグの種類によっては終了タグが存在しないものもあります。そのような要素は「空要素」と呼びます。空要素の場合、スラッシュはつけてもつけなくても構いません。

文法　空要素の記述方法

```
<開始タグ>
```

例　空要素の記述例

```
<br>
```

また、各種タグはそれぞれ異なる「属性」を持っています。属性とは、タグにつける付加情報のことで、例えばリンクタグであればリンク先のURLなどを指定します。また、次章で学ぶCSSを一部のタグにだけ適用する場合や、JavaScriptで特定のタグを操作する場合などにもあらかじめ属性をつけておきます。属性は開始タグに記述します。1つのタグに対して複数の種類の属性をつけることが可能です。

文法　属性の記述

```
<開始タグ 属性1="値" 属性2="値">内容</終了タグ>
```

例　属性の記述例

```
<a href="top.html">TOPページへ</a>
```

属性値はダブルクォート(")のかわりにシングルクォート(')で囲んでも構いません。また、属性の指定順序に決まりはありません。

HTMLの構成

HTML文書は、いくつものタグを組み合わせて構成します。記述する際は、要素の中に別の要素を入れ込んでいく構造(入れ子構造またはネスト構造と呼ぶ)にします。この時注意しなければならないのは、終了タグの位置です。

例　良い例

```
<div><p>これは段落です。</p></div>
```

例　悪い例

```
<div><p>これは段落です。</div></p>
```

タグが交差するように配置してはいけません。必ず1つのタグを包むように配置していきます。

HTML の例

ここでは、第 1 章で作成したアプリのソースコードを例に解説します。

解説

index.html

```
<!DOCTYPE HTML>      ……①
<html>               ……②
<head>               ……③
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
    scale=1, maximum-scale=1, user-scalable=no">
    <meta http-equiv="Content-Security-Policy" content="default-src
    *; style-src * 'unsafe-inline'; script-src * 'unsafe-inline'
    'unsafe-eval'">                                              ……④
    <script src="components/loader.js"></script><script src="classic.js"></script>
    ……⑤
    <link rel="stylesheet" href="components/loader.css">
    <link rel="stylesheet" href="css/style.css">                 ……⑥
    <script>
       alert("こんにちは");                                      ……⑦
    </script>
</head>
<body>          ……⑧
    はじめてのプログラム
</body>
</html>
```

ここで登場しているのは、アプリを作る上で最低限必要となるタグです。この本の学習範囲内では、基本的に <script> タグや <body> タグの中以外を変更する必要はありませんので、それ以外のタグは削除しないでください。以下に各タグの意味を解説します。

①<!DOCTYPE HTML>

HTMLの最新バージョンで記述された文書であることを表すタグです。終了タグはありません。

②<html>

HTML文書であることを表すタグです。文書全体をこのタグで囲みます。

③<head>

文書全体に関する情報を定義するタグです。このタグ自体はあまり意味を持たず、中に入っているタグがさまざまな意味を持ちます。

④<meta charset="utf-8">

<meta>タグは、メタ情報と呼ばれるHTML文書の補足情報を持つタグです。文書がどの文字コードで書かれているかといった情報や、スマートフォンなどの小さいサイズのスクリーンで見たときに拡大・縮小する設定などを指定します。

⑤<script src="components/loader.js"></script><script src="classic.js"></script>

<script>タグはJavaScriptのプログラムを記述するためのタグです。また、JavaScriptで記述されたファイルを読み込むこともできます。

loader.jsはMonacaでアプリ開発を行う場合に必要なファイルなので、消さないようにしましょう。classic.jsはエラーをプレビュー表示するために必要なファイルなので消しても問題ありませんが、あった方が便利です。

⑥<link rel="stylesheet" href="components/loader.css">

<link rel="stylesheet" href="css/style.css">

<link>タグは外部ファイルを読み込むタグです。ここではCSSで記述されたファイルを読み込んでいます。loader.cssはMonacaでアプリ開発を行う際に必要なファイルなので、消さないようにしましょう。style.cssは3章以降で解説があります。

⑦<script> alert("こんにちは"); </script>

<script>タグでJavaScriptのプログラムを記述しています。

⑧<body>

<body>タグは本文を記述するためのタグです。文章や画像などを記述します。

ポイント　用語説明：ソースコード

ソースは「元となるもの」、コードは「命令文」のことを意味します。ここでは、画面の元となっている命令文、という意味で使っています。「ソース」と「コード」はそれぞれ単独で使う場合もあります。

ポイント　用語説明：文字コード

コンピューターで利用される文字は、内部的には番号で管理されています。文字と番号の対応関係のことを文字コードといいます。世界各国の文字を表現するためにさまざまな種類の文字コードが開発されていて、よく使われる文字コードに「UTF-8」「Shift_JIS」「EUC-JP」等があります。最新の HTML では文字コードに UTF-8 を使用することが推奨されています。HTML ファイルを保存するときに文字コードの指定を忘れないようにしましょう。

<body>要素内に記述する要素の種類

アプリの画面を作成する場合は、文章や画像などを表示するために各種要素を <body> 要素の中にタグで記述していきます。HTML では非常に多くの要素が用意されていますので、ここでは主要なものを紹介します。

終了タグのある要素

要素名	概要
h1	見出しを定義します。h1 ～ h6 まであり、h1 が最も高レベル、h6 が最も低レベルな見出しです。 例： 見出し
p	文章の段落を定義します。 例： 　　文章の段落を定義します。
div	特に意味を持たないタグです。複数のタグをまとめて扱うときや、四角い枠を描画したいときに使います。 例： 　　 見出し 　　 段落
a	リンクを定義します。 href 属性・・・リンク先の URL を指定します。 例：TOP へ
button	ボタンを定義します。 例：<button> ボタン </button>

空要素（終了タグのないタグ）

要素名	概要
img	画像を参照します。 src 属性・・・画像の参照先を指定します。 alt 属性・・・画像が何らかの理由で表示できなかった場合に、画像の変わりに表示する文字列を指定します。 例：<img src="flower.jpg" alt=" 花 ">

すべての要素につけられる属性

属性名	概要
id	要素を識別するための ID です。文書内で重複する値を指定することはできません。 例： …
class	CSS のクラス名を指定します。（→第 3 章） 例： …

次節では、これらの中でも特に頻繁に使われるリンクと画像の表示についてサンプルを挙げて解説します。

リンク

リンクは、ある画面から別の画面へ移動する機能です。リンクを設定する際に重要となるのが、パスという考え方です。パスとは、HTMLなどのファイルが存在しているコンピューター上の住所のことです。パスの指定方法には絶対パス指定と相対パス指定の2通りがあります。

文法　リンクの設定

```
<a href="リンク先のパス">リンク文字列</a>
```

絶対パス指定

パスを全て記述する方法です。Windowsパソコンの「ドキュメント」フォルダ内に入っているsample.txtというファイルを表す場合は、「C:¥Users¥ユーザー名¥Documents¥sample.txt」がパスになります。また、Webサイトの場合はWebブラウザ上部のアドレスバーに表示される、「https://」から始まる文字列が絶対パスになります。アプリ内にインターネット上のWebサイトを表示する場合などは、こちらの方法を使います。

実習

本章のひな形プロジェクトを開き、index.htmlの14行目に以下の要素を追記しましょう。

```
<a href="https://edu.monaca.io/">Monacaへ</a>
```

サンプルプログラム

index.html

```
1  <!DOCTYPE HTML>
2  <html>
3  <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1, maximum-
   scale=1, user-scalable=no">
```

```
6     <meta http-equiv="Content-Security-Policy" content="default-src *; style-
    src * 'unsafe-inline'; script-src * 'unsafe-inline' 'unsafe-eval'">
7     <script src="components/loader.js"></script><script src="classic.js"></
    script>
8     <link rel="stylesheet" href="components/loader.css">
9     <link rel="stylesheet" href="css/style.css">
10    <script>
11    </script>
12  </head>
13  <body>
14    <a href="https://edu.monaca.io/">Monaca へ </a>
15  </body>
16  </html>
```

「Monaca へ」というリンク文字列をタップすると、以下のような画面になるはずです。

実行結果

外部 Web サイトをアプリ内で開けたことが確認できました。

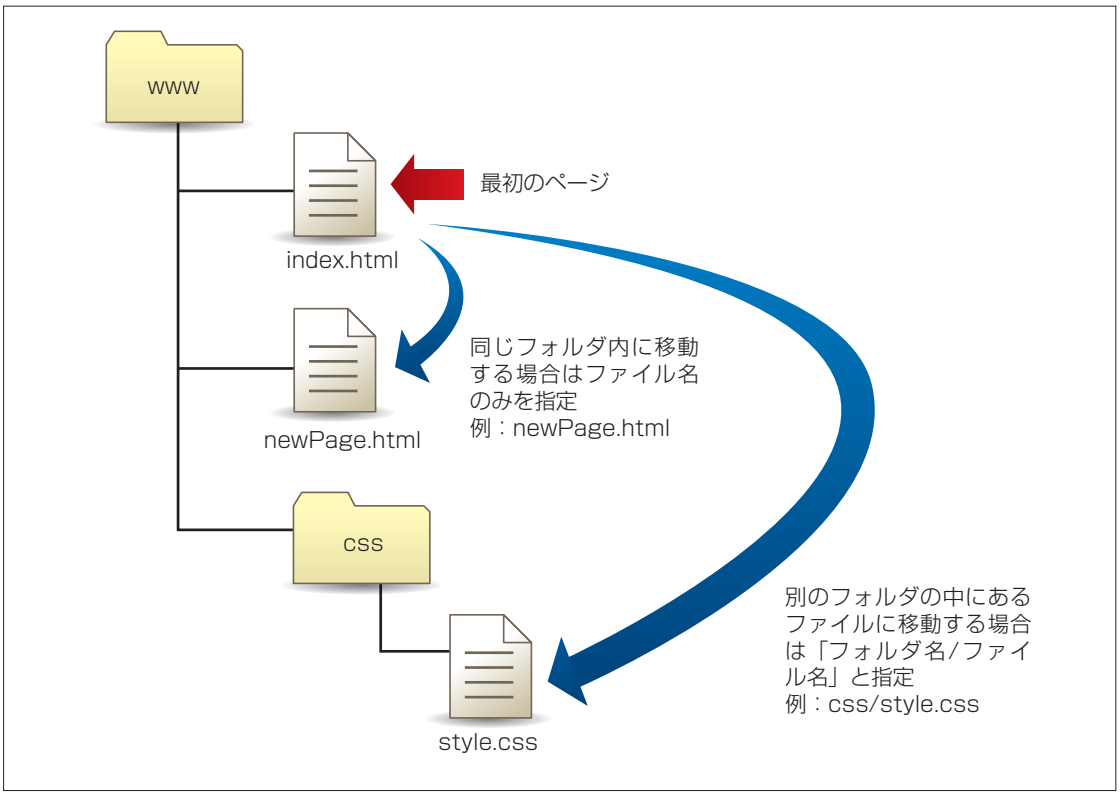
なお、リンク先によってはプレビュー機能では確認できない場合があります。その場合は、「Web 公開機能」や「Monaca for Study」アプリを使って確認してみましょう。

相対パス指定

現在のファイルから見た、対象ファイルまでの位置を指定する方法です。サイトやアプリ内の別のHTMLファイルに移動する場合には、こちらの方法を使用します。相対パス指定する場合、記述方法には以下の決まりがあります。

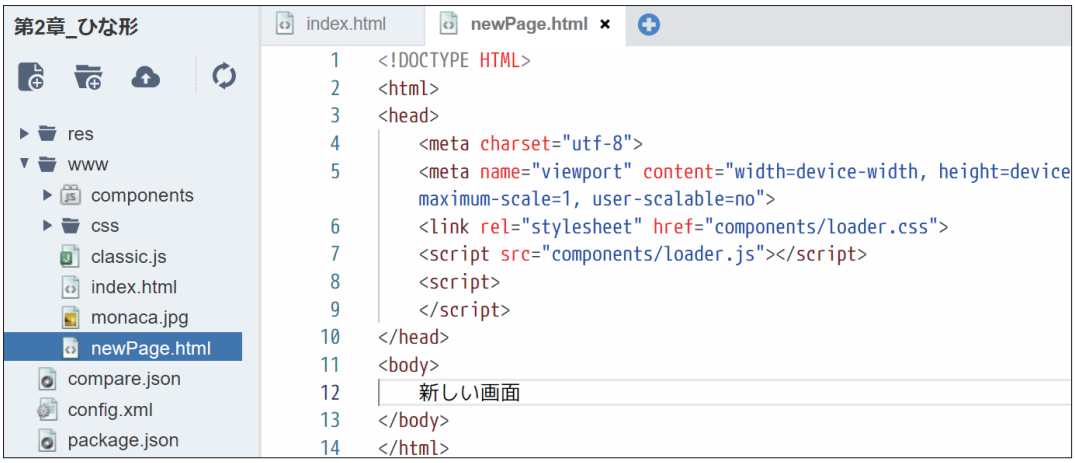
・同一フォルダ内のページに移動する場合はファイル名の指定だけで良い。
・フォルダとフォルダの区切り、またはフォルダとファイルの区切り文字として、フォルダ名の後ろにスラッシュをつける。
・一つ上のフォルダは .. という記号で表す。

相対パスの指定方法を以下の図で示します。



実習

今まで記述していたファイルは index.html という名前のファイルですが、このファイルはアプリ起動時に最初に表示されるファイルです。ここから別の HTML ファイルに移動してみましょう。移動先は [www] フォルダ直下にあらかじめ配置されている、newPage.html です。このファイルをダブルクリックして開くとわかりますが、「新しい画面」という文字列が <body> タグの中に記述されています



それでは、index.html のタブに戻って 15 行目に、以下の要素を追記してください。

```
<a href="newPage.html">次の画面へ</a>
```

サンプルプログラム

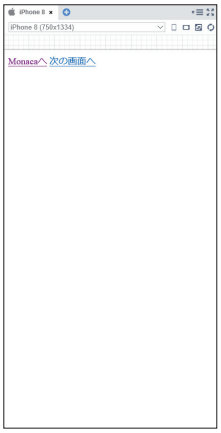
index.html

```
1  <!DOCTYPE HTML>
2  <html>
3  <head>
4      <meta charset="utf-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1,
   maximum-scale=1, user-scalable=no">
6      <meta http-equiv="Content-Security-Policy" content="default-src *; style-
   src * 'unsafe-inline'; script-src * 'unsafe-inline' 'unsafe-eval'">
7      <script src="components/loader.js"></script><script src="classic.js"></
   script>
8      <link rel="stylesheet" href="components/loader.css">
9      <link rel="stylesheet" href="css/style.css">
10     <script>
11     </script>
```

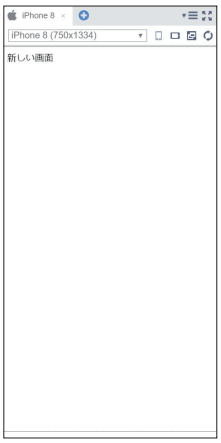
```
12  </head>
13  <body>
14      <a href="https://edu.monaca.io/">Monaca へ </a>
15      <a href="newPage.html"> 次の画面へ </a>
16  </body>
17  </html>
```

同じ [www] フォルダ内の HTML ファイルに移動するので、相対パス指定でリンク先を指定しています。　ここまで出来たら、プレビュー画面または Monaca デバッガーで実行してみましょう。

実行結果



[次の画面へ] をクリックまたはタップすると、newPage.html へ移動します。



このようにして、リンクを使って画面の切り替えを行うことができるようになります。

画像の表示

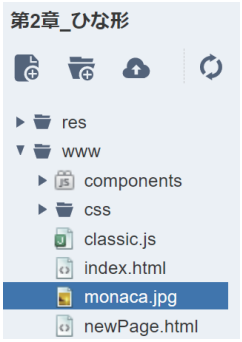
画面上に写真やイラストなどの画像を表示するには、HTML ファイルから画像ファイルを参照するようにタグで指定します。画像ファイルのパス（画像が置いてある場所）はリンクと同様、絶対パスまたは相対パスで指定します。

文法　画像の表示

```
<img src="画像ファイルのパス">
```

実習

表示する画像は、[www] 直下に配置されている monaca.jpg を利用します。



index.html の 16 行目に、以下の要素を追記してください。

```
<img src="monaca.jpg">
```

サンプルプログラム

index.html

```
1  <!DOCTYPE HTML>
1  <html>
2  <head>
3      <meta charset="utf-8">
4      <meta name="viewport" content="width=device-width, initial-scale=1,maximum-
5  scale=1, user-scalable=no">
```

```
6     <meta http-equiv="Content-Security-Policy" content="default-src *; style-
    src * 'unsafe-inline'; script-src * 'unsafe-inline' 'unsafe-eval'">
7     <script src="components/loader.js"></script><script src="classic.js"></
    script>
8     <link rel="stylesheet" href="components/loader.css">
9     <link rel="stylesheet" href="css/style.css">
10    <script>
11    </script>
12  </head>
13  <body>
14    <a href="https://edu.monaca.io/">Monaca へ </a>
15    <a href="newPage.html"> 次の画面へ </a>
16    <img src="monaca.jpg">
17  </body>
18  </html>
```

実行結果

このように、HTML は画面上に表示したい文字列やリンク、画像などを指定するために利用します。次章では表示した内容にデザインを適用し、見栄えを良くする方法を学んでいきます。

第 3 章

CSS 入門

前章で学んだ HTML は、画面に表示する内容を定義するための技術でした。本章で学ぶ CSS を HTML に組み込んで使うことで、画面を装飾することができます。

> 前章で作成したプロジェクトを継続して使用するか、サポートページから本章のひな形をインポートしてください。

CSSとは

CSS（Cascading Style Sheets）は、HTML文書を装飾するための技術です。背景や文字の色設定を行ったり、文字や画像のサイズ、表示位置を調整したりと、画面にデザインを適用するために使われます。 色やサイズなどの一つ一つのデザインのことを「スタイル」と呼びます。「Cascading」は重ね合わせるといった意味がある言葉ですので、CSS（Cascading Style Sheets）はたくさんのスタイルを重ね合わせてデザインを完成させるための文書、という意味になります。

CSS を HTML ファイルに読み込む方法

CSSは、HTMLファイルの中に読み込む形で利用します。まず、CSSのコードのみをファイルに記述し、拡張子を.cssとして保存します。 次に、HTMLファイルに<link>タグを記述し、href属性にCSSファイルのパスを指定します。

文法　CSS ファイルの読み込み

```
<link rel="stylesheet" href="CSSファイルのパス">
```

なお、本章のひな形プロジェクトやクラシック・テンプレートのindex.htmlには、あらかじめ「style.css」という名前のCSSファイルを組み込む<link>タグが記述されています。

index.html

```
1   <!DOCTYPE HTML>
2   <html>
3   <head>
4       <meta charset="utf-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=
6       <meta http-equiv="Content-Security-Policy" content="default-src * data: gap: content: https://ssl.g
7       <script src="components/loader.js"></script>
8       <link rel="stylesheet" href="components/loader.css">
9       <link rel="stylesheet" href="css/style.css">
10      <script>
11      </script>
12  </head>
```

CSS の書き方

CSS を記述する際には「どの要素に対して」「どのようなスタイル」を適用するのか、の 2 つの情報が必要です。「どの要素に対して」は「セレクタ」という仕組みでタグなどを指定します。また「どのようなスタイル」は「プロパティ」と「値」で指定します。プロパティには文字の色や背景色またサイズなどさまざまなものが用意されており、適応させたい色や数値を値として指定できます。

文法　セレクタとプロパティの記述方法

```
セレクタ {
        プロパティ : 値;
        プロパティ : 値;
        ～
}
```

例　セレクタとプロパティの記述例

```
p {
        color: red;
        font-size: 10px;
}
```

この例では、HTML 文書内の <p> タグに対して、文字色を赤に、フォントサイズを 10px にする、という指定を行っています。

ポイント　用語説明：px（ピクセル）

コンピューター上に表示される写真や図形などは、ピクセルという点の集合によって描画されています。点 1 つが 1px です。一般的にアプリの画面を作るときにはこの px という単位を利用します。

セレクタの種類

セレクタは対象要素を指定する方法で、複数の種類があります。タグ名が同じ要素すべてにスタイルを適用したい場合と、1 つの要素に対してのみスタイルを指定したい場合とでは、利用するセレクタが異なります。状況に応じて適切なセレクタを選択しましょう。

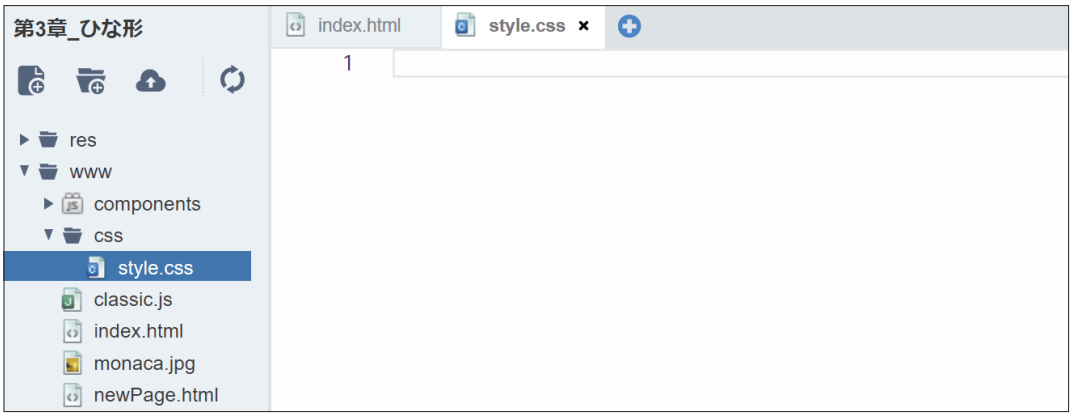
・タグセレクタ
　対象要素をタグ名で指定します。
・ID セレクタ
　対象要素を ID 属性値で指定します。ID 属性値は HTML 文書の中で一意となる（重複する値を設定できない）ので、特定の要素 1 つだけにスタイルを適用したい場合に利用します。
・クラスセレクタ
　CSS におけるクラスとは、スタイルをひとまとめにして名前をつけたもののことです。クラスはどのタグに対しても付けることができるので、複数の要素の中から任意の要素を選択してスタイルを適用したい場合に利用します。

各セレクタの記述方法

セレクタ	書き方	例
タグセレクタ	タグ名 {…}	p {…}
ID セレクタ	#ID {…}	#id1234 {…}
クラスセレクタ	. クラス名 {…}	.className {…}

実習

本章のひな形、もしくは前章で作成したプロジェクトを開き、[css] フォルダ内の style.css を開きます。



style.css は、はじめは空の状態になっています。最初に、タグセレクタで <a> タグすべてに対してスタイルを適用してみます。以下のコードを記述してください。

サンプルプログラム

style.css

```
1  a {
2      font-size: 30px;
3  }
```

プレビュー画面で確認すると、リンク文字列の大きさが変更されていることがわかります。 これがタグセレクタによる指定です。

実行結果

続いて、ID セレクタを試してみましょう。

index.html　14 行目の、「Monaca へ」リンクを設定している <a> タグに、ID 属性を指定します。

サンプルプログラム

index.html

```
14      <a href="https://edu.monaca.io/" id="target">Monaca へ </a>
```

次に style.css を開き、先ほど記述した箇所の下に以下のコードを追記します。

サンプルプログラム

style.css

```
5  #target {
6      color: red;
7  }
```

ここまでをプレビュー画面で確認すると、「Monacaへ」というリンク文字列が赤色で表示されます。

実行結果

最後に、クラスセレクタによる指定方法を試してみましょう。 index.html 15行目の、「次の画面へ」リンクを設定している<a>タグに、class属性を指定します。

サンプルプログラム

index.html

```
15      <a href="newPage.html" class="bright">次の画面へ</a>
```

次に style.css を開き、先ほど記述した箇所の下に以下のコードを追記します。

サンプルプログラム

style.css

```
9  .bright {
10     background-color: yellow;
11 }
```

background-color は、背景色を指定するプロパティです。
プレビュー画面で結果を確認すると、背景が黄色で表示されます。

実行結果

CSSのクラスは、複数のタグに対して同じ属性値を設定することができます。
試しに、<body>内の他のタグにも「class=”bright”」という属性を追加してみましょう。属性を追加したタグの背景色がすべて黄色くなることが確認できると思います。

プロパティの種類

CSSのプロパティは膨大な種類がありますので、ここでは特によく使われるプロパティを紹介します。

色を指定するプロパティ

プロパティ	説明	例
color	文字色を設定します。	color: red;
background-color	背景色を設定します。	background-color: red;
border	線の色（および線種と線の太さ）を設定します。	border: solid 1px red; 線種、線の太さ、線の色の順に設定します。 solid は直線を表します。

カラーコード

色の表現方法は、「red」や「blue」などの色の名称を指定する方法の他に、カラーコードと呼ばれる方法があります。コンピューターのディスプレイに表示される色は、光の三原色（赤、緑、青）を混ぜ合わせて作られています。　それぞれの色の含有量を最小0から最大255までの数値で表し、16進数にして並べた6桁の数値がカラーコードです。カラーコードの先頭には#（シャープ）を付けて記述します。

カラーコードの例

```
#ff00ff
```

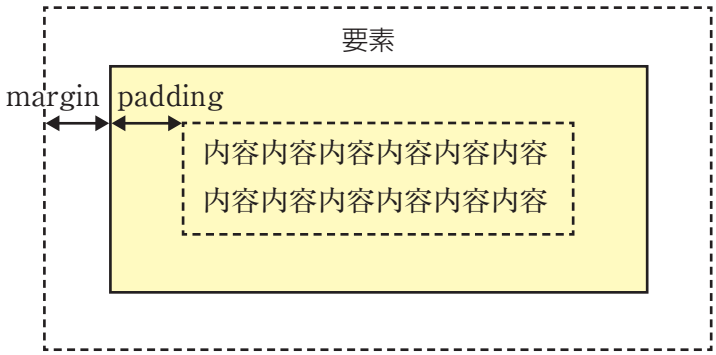
赤がff（255）、緑が00（0）、青がff（255）なので、原色の赤と青を混ぜた色＝紫になります。

カラーコードを調べるときは一般的にPhotoshop、Illustratorなどのデザインやイラスト制作用のグラフィックソフトを使います。グラフィックソフトが無い場合は、カラーコードを算出するWebサービスが多数公開されていますので、各自調べてみましょう。

サイズや位置を指定するプロパティ

プロパティ	説明	例
font-size	文字のサイズを設定します。	font-size: 12px;
text-align	要素内の横方向の配置を設定します。	text-align: left;（左寄せ） text-align: right;（右寄せ） text-align: center;（中央揃え） text-align: justify;（均等割付）
width	要素の横幅を設定します。	width: 100px;
height	要素の高さを設定します。	height: 300px;
margin	枠線の外側の余白を設定します。	margin: 20px;
padding	枠線の内側の余白を設定します。	padding: 10px;

marginとpaddingはどちらも余白の幅を指定するプロパティですが、余白を取る位置が異なります。以下の図に示すように、marginが枠線の外側、paddingが枠線の内側の余白になります。



marginプロパティとpaddingプロパティは、ハイフン（-）に続けて方向を表す単語を付けると、一辺に対してのみ余白を設定することができます。例えば、margin-topは外側の上余白を設定します。

方向	外側余白	内側余白
上	margin-top	padding-top
下	margin-bottom	padding-bottom
左	margin-left	padding-left
右	margin-right	padding-right

なお、サイズや位置を指定する単位は、px（ピクセル）の他に、%（パーセント）もよく利用されます。% 指定の場合は、画面全体または外側にあるタグを 100% として計算されます。

実習

プロジェクトの style.css に、以下のコードを追加します。

サンプルプログラム

style.css

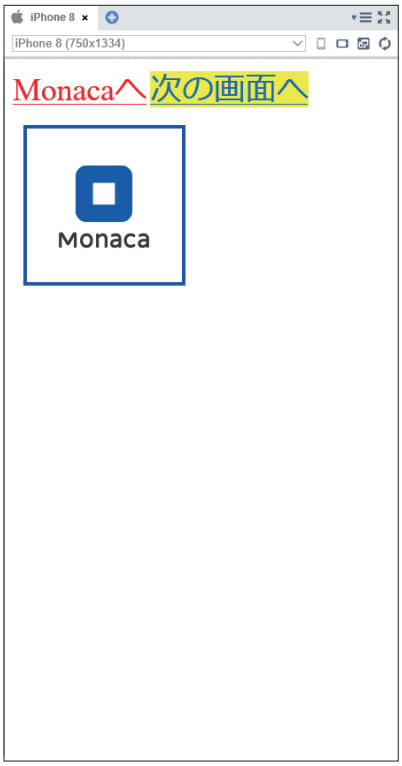
```
13 img {
14     width: 30%;
15     border: solid 3px #0000ff;
16     margin: 10px;
17     padding: 20px;
18 }
```

1 つ目の width プロパティは、要素の横幅を設定するプロパティです。画像の幅を画面全体に対して 30% に設定していますので、画像が小さく表示されています。

2 つ目の border プロパティは、要素の周囲に線を表示します。カラーコード「#0000ff」は、青を表しますので、3px の太さの青い枠線が文字列の周囲に出現します。

残りの margin プロパティと padding プロパティは、余白を設定するプロパティです。枠線の外側に 10px、枠線の内側に 20px の余白が作られています。

実行結果



このように、さまざまなCSSプロパティを組み合わせることで、アプリの画面を自由自在にデザインすることができます。

4

第4章

JavaScript 入門

本章からはいよいよアプリを動かすための技術、JavaScriptを学んでいきます。JavaScriptは、主にWebページやモバイルアプリの画面上の部品を操作するために利用されます。なおJavaScriptと似たような名前のプログラミング言語に「Java」というものがありますが、これはJavaScriptの略ではありません。JavaとJavaScriptは違う言語です。JavaScriptを省略して呼ぶ場合はJS(ジェイエス)と呼びます。

クラシック・テンプレートを使用するか、サポートページから本章のひな形をインポートしてください。

JavaScriptの書き方

第 1 章では以下の JavaScript を実行しました。繰り返しになりますが、JavaScript は HTML ファイルの <script> タグの中に記述します。

解説

```
10    <script>
11        alert(" こんにちは ");
12    </script>
```

CSS ファイルのように、JavaScript だけを記述したファイル(拡張子には「.js」が付きます)を作っておいて、HTML ファイルに読み込む方法を取ることもできます。外部の JavaScript ファイルを HTML に読み込むには、以下のように記述します。

文法　JavaScript ファイルの読み込み

```
<script src="JavaScript ファイルのパス "></script>
```

書き方のルール

まずは JavaScript を記述するにあたって必ず守らなければならないルールを理解しましょう。

- 基本的に半角の英数字と記号のみを使う。
- シングルクォート (') とダブルクォート (") で括られた範囲内では全角文字を利用することもできる。
- 大文字と小文字は別の文字として扱われる。
- 命令文の末尾にはセミコロン (;) をつける。
- 複数行に渡るひとまとまりの命令群を波かっこ { } で囲む。囲まれた範囲をブロックと呼ぶ。

また、上記のルールを守っていれば、JavaScript のコードは自由に改行や半角スペースなどを挿入して良いことになっています。例えば、以下の 2 つのコードを見比べて見てください。

例　プログラムA

```
for(i=0;i<10;i++){alert(i);}
```

例　プログラムB

```
for(i = 0; i < 10; i++) {
    alert(i);
}
```

この2つは、どちらも全く同じことが書かれています。しかし、プログラムAは見づらく、プログラムBは見やすいと感じられるのではないでしょうか。
このように見やすいコードを書くためのポイントは2つです。

- 単語や記号の間には半角スペースを入れる。
- インデントを正しく設定する。

インデントとは

インデントというのは、文章を記述する際に空白スペースやタブなどを用いて見やすいように字下げを行うことです。プログラムBでは、2行目のalert(i); という命令文が少し右にずれた位置から開始されています。これがインデントを設定した状態です。

プログラムBでは1行目で波かっこが開始されていて、3行目で波かっこが終了しています。JavaScriptでは波かっこを多用するのですが、プログラム中にかっこがたくさん出てくると、開始かっこに対応する終了かっこが見つけづらくなってきます。
そこでかっこの開始行と終了行を同じ横位置に揃え、かっこの中は右にずらして記述することで、かっこの対応関係が一目でわかるようになります。

インデント処理を行うには、キーボードの[Tab]キーを一度押します。Monaca Educationの標準設定では半角スペースが4つ分、挿入されます。もちろん、半角スペースを直接入力しても問題ありません。

```
for(i = 0 ; i < 10; i++) {
    alert(i);
}
```

コメント

alert 命令などの先頭にスラッシュ（/）を 2 つ付けると命令は無効化されます。

文法　一行のコメント

```
//alert("こんにちは");
```

JavaScript では、// 以降の文字列はコメント（プログラムの実行に影響を与えないメモ書き）となります。

なお、複数行にわたる文字列をコメントにする場合は、/* と */ でコメントする範囲を囲みます。

文法　複数行のコメント

```
/*
コメントとして記述した内容は、
スクリプトには影響しません。
*/
```

コメントには、自分で後からプログラムを読み返したときや、だれか他の人がそのプログラムを見たときに、処理内容の理解を助ける説明文を記述します。例えば、「//ラジオボタン A が選択された場合は登録処理を行う」といった具合です。
また、今は使わないけれど消去したくはない、残しておきたいコードをコメントにして無効化する場合もあります。

HTML のコメントは <!-- -->でしたが、JavaScript のコメントとは記述方法が異なるので混同しないように気を付けましょう。

データの扱い方

はじめに覚えなければならないのは、JavaScriptで文字列や数値などのデータを扱う方法です。皆さんが考え事をするとき、頭の中にいろいろな物事(データ)を思い浮かべますね。それと同じように、コンピューターがプログラムを実行するときには、メモリという装置上にたくさんのデータを記憶します。メモリ上にデータを記憶するには、まずデータの入れ物を用意しなければなりません。この入れ物のことを変数と呼びます。

変数の作り方

メモリ上に変数を作る作業を、変数の「宣言」と言います。変数に名前を付けて、「この名前の変数を今から使いますよ」ということをコンピューターに宣言しておくのです。データを扱う前には必ず変数の宣言を行います。

文法　変数宣言の書式

```
let 変数名;
```

例　xという名前の変数を作る

```
let x;
```

変数名には自由な名前を付けることができますが、読みやすいプログラムにするためには何のデータを入れるための変数なのかを推測しやすい名前にしましょう。例えば金額のデータを入れる変数であれば、「money」や「price」といった変数名が良いでしょう。

変数の使い方

変数を作った直後は、まだ変数の中には何もデータが入っていない、空っぽの状態になっています。変数にデータを入れるには、以下のようにします。

文法　変数へ値を入れる

```
変数名 = 値;
```

例　変数 x の中に「10」という数値を入れる

```
x = 10;
```

例　変数 x の中に「こんにちは」という文字列を入れる

```
x = "こんにちは";
```

※ JavaScript で文字列データを扱う場合は、ダブルクォート (") またはシングルクォート (') で囲みます。

注意しなければならないのは、この「=」記号は算数の「=」記号とは意味合いが異なるということです。算数では左右の値が等しいということを意味しますが、JavaScript のイコール記号は右辺の値を左辺に入れる、という意味になります。左右が逆になってしまうと正しく動きませんので注意しましょう。

なお、変数へ値を入れることを、値の「代入」といいます。以降も頻出する言葉なので覚えておきましょう。

宣言と代入は、1 行にまとめて同時に行うこともできます。

文法　宣言と代入を同時に行う

```
let 変数名 = 値;
```

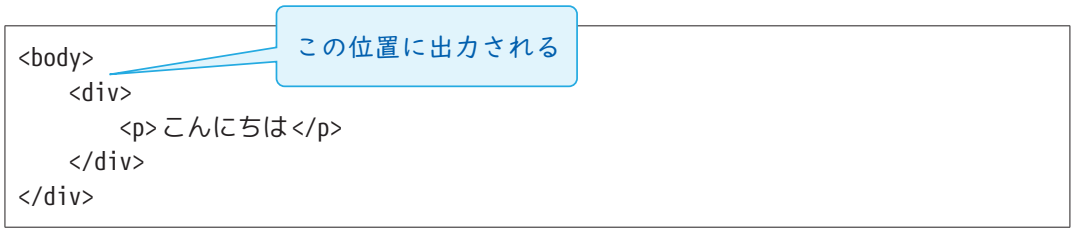
JavaScriptから画面にデータを出力する

データをアプリの画面に表示するにはさまざまな方法がありますが、ここでは最も簡単な命令を使って確認してみましょう。

文法　<body>タグ内にデータを出力する

```
document.write(表示するデータ);
```

この命令を使うと、<body>タグ内の一番上の位置にデータを書き込みます。



実習

本章のひな形プロジェクトを開き、index.html を編集しましょう。

まずは <body> タグ内に「今日も一日がんばりましょう。」という文字列を記述します。

サンプルプログラム

index.html

```
15      <body>
16          今日も一日がんばりましょう。
17      </body>
```

続いて、<script> タグ内に JavaScript による命令を記述します。

サンプルプログラム

index.html

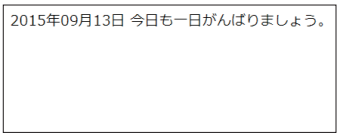
```
10      <script>
11          let today = "2015年09月13日";
12          document.write(today);
13      </script>
```

サンプルプログラム

index.html（完成版）

```
<!DOCTYPE HTML>
<html>
<head>
   <meta charset="utf-8">
   <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no">
   <meta http-equiv="Content-Security-Policy" content="default-src *; style-src * 'unsafe-inline'; script-src * 'unsafe-inline' 'unsafe-eval'">
    <script src="components/loader.js"></script><script src="classic.js"></script>
   <link rel="stylesheet" href="components/loader.css">
   <link rel="stylesheet" href="css/style.css">
   <script>
        let today = "2015年09月13日";
        document.write(today);
   </script>
</head>
<body>
    今日も一日がんばりましょう。
</body>
</html>
```

実行結果



このままでは<body>タグに文字列を記述したのと変わらないので、次は自動的に現在の日付が表示されるように変更してみましょう。

今日の日付を取得する

JavaScriptには、さまざまなデータを扱うための便利な命令群があらかじめ用意されています。そして、命令にはたくさんの種類があるので、カテゴリごとに分けられています。日付を扱うための命令は、「Date」の中に含まれています。

日付に関する操作

日付を扱う命令を使えるようにするための準備

```
let 変数 = new Date();
```

年を取得する命令

```
変数.getFullYear();
```

月を取得する命令

```
変数.getMonth();
```

※現在の月から1引いた値が取得される（現在1月なら、0という値が取得される）

日を取得する命令

```
変数.getDate();
```

時間を取得する命令

```
変数.getHours();
```

分を取得する命令

```
変数.getMinutes();
```

秒を取得する命令

```
変数.getSeconds();
```

曜日を取得する命令（日曜日～土曜日まで表す、0 ～ 6 の数値を返す）

```
変数.getDay();
```

これらの命令を使って、アプリに現在の日付を表示します。

実習

プロジェクトの <script> タグ内を以下のように変更してください。

サンプルプログラム

index.html

```
10      <script>
11          // 日付に関する命令を使えるようにする
12          let date = new Date();
13          // 年、月、日の取得
14          let year = date.getFullYear();
15          let month = date.getMonth() + 1;
16          let day = date.getDate();
17          // 日本の表記にする
18          let today = year + "年" + month + "月" + day + "日";
19          document.write(today);
20      </script>
```

完成したら、プレビュー画面で今日の日付が出力されていることを確認して下さい。

少し難しい内容になりましたので、順番に処理を追っていきましょう。
まず、12行目で日付を扱う命令の利用準備を行っています。日付関連の命令を使う場合は、最初にこの処理が必要となります。
14 ～ 16行目では、現在の年、月、日を取得します。月を取得する命令だけは少し特殊で、現在の月から1引いた値が取得されてしまうので、1を加算して正しい月に変換しています。この場合の「+」記号は、算数と同じで足し算をするという意味です。
最後に、18行目で取得した年、月、日を日本表記の日付形式にしています。ここでも「+」記号が出てきていますが、数値ではなく文字列を「+」記号で繋いでいるので、計算処理はできません。文字列を「+」記号で繋いだ場合は、足し算ではなく文字列の連結になります。

ポイント　注意！「+」記号には二通りの意味がある

> 数値同士を「+」記号で繋いだ場合は足し算、文字列を「+」記号で繋いだ場合は文字列の連結となる

これで、今日の日付を表示するアプリは完成です。明日以降、またこのアプリを実行してみて下さい。日付が更新されていることが確認できるはずです。

このように、JavaScriptを使うことで、いつ見ても同じ画面ではなく、状況に応じて異なる結果を表示することができるようになります。

5

第 5 章

条件分岐

ここまでに実行したプログラムは、上から順番にすべての処理が実行されていました。しかしこれだけでは実用的なアプリは作れません。例えばゲームではよく、選択肢に応じてその後のシナリオが変化する、といった仕組みがあると思います。そのようにある条件に応じてその後行われる処理を振り分けることを条件分岐といい、アプリ開発には欠かせない考え方です。
本章ではアプリを発展させて、曜日ごとにメッセージが変わるアプリを作成し、条件分岐を実現する方法を学びます。

前章で作成したプロジェクトを継続して使用するか、サポートページから本章のひな形をインポートしてください。

if文

JavaScriptで条件分岐を実現するには、if文という命令を使います。if文を利用すると、さまざまな条件によって異なる処理を実行することができるようになります。

文法　if文の書き方

```
if（条件式）{
    条件式が正しい場合に実行する処理
} else {
    条件式が正しくない場合に実行する処理
}
```

ある条件を指定して、その条件が正しかった場合と正しくなかった場合とで、処理を2方向に分岐します。else以降は省略が可能なので、正しくなかった場合は何もしない、という動作をさせることもできます。

条件式の書き方

if文で条件を指定するための「条件式」は、比較演算子という記号を使って記述します。ある２つの値に対して、「等しい／等しくない／小さい／大きい／以下／以上」のうちいずれかの比較を行います。

比較演算子の種類

演算子	概要	条件式の例	結果
==	左辺と右辺が等しい場合は正しい	1 == 1	正しい
!=	左辺と右辺が等しくない場合は正しい	1 != 2	正しい
<	左辺が右辺より小さい場合は正しい	1 < 1	正しくない
<=	左辺が右辺以下の場合は正しい	1 <= 1	正しい
>	左辺が右辺より大きい場合は正しい	1 > 1	正しくない
>=	左辺が右辺以上の場合は正しい	1 >= 1	正しい

特によく使うのは、２つの値が等しいかどうかを判定する「==」演算子です。イコール記号を２つ繋げて記述します。

実習

本章のひな形、もしくは前章で作成したプロジェクトを開き、<script>タグの下に以下の処理を追記します。

```
// 曜日を取得
let weekday = date.getDay();
if(weekday == 0) {
    document.write(" 今日は日曜日です ");
} else {
    document.write(" 今日は日曜日ではありません ");
}
```

サンプルプログラム

index.html

```
10      <script>
11          // 日付に関する命令を使えるようにする
12          let date = new Date();
13          // 年、月、日の取得
14          let year = date.getFullYear();
15          let month = date.getMonth() + 1;
16          let day = date.getDate();
17          // 日本の表記にする
18          let today = year + "年" + month + "月" + day + "日";
19          document.write(today);
20
21          // 曜日を取得
22          let weekday = date.getDay();
23          if(weekday == 0) {
24              document.write("<br>今日は日曜日です");
25          } else {
26              document.write("<br>今日は日曜日ではありません");
27          }
28      </script>
```

実行結果を確認して、曜日が正しく判定されているか見てみましょう。

実行結果

今日の曜日を取得するには、Date の getDay という命令を使います。この命令は、曜日を数値に置き換えた結果を取得します。曜日と数字の対応表は下記を参考にしてください。

getDayが取得する値

曜日	数値
日曜日	0
月曜日	1
火曜日	2
水曜日	3
木曜日	4
金曜日	5
土曜日	6

ソースコード23行目のif文で、今日の曜日を表す数値と、日曜日を表す0が等しいかどうか比較をしています。もし日曜日であれば等しいので、if文の下の「今日は日曜日です」というメッセージを表示する処理を実行します。もし日曜日以外であれば、等しくないという比較結果になるので、else文の下の「今日は日曜日ではありません」が表示されます。

多方向分岐

ここまででではまだ日曜日かどうかの判定しかできていないので、曜日ごとに異なるメッセージを表示してみましょう。先ほどのif文では正しいか、正しくないかの判断を行って処理を２方向に分けていましたが、曜日ごとにメッセージを変えるには７方向に分岐をさせなければなりません。このような場合はelse if文を使います。

文法　else if文

```
if (条件式1) {
    条件式1が正しい場合に実行する処理
} else if(条件式2) {
    条件式2が正しい場合に実行する処理
} else {
    条件式が正しくない場合に実行する処理
}
```

else if文は、if文とelse文の間に記述します。else if文は何回でも指定することができるので、必要な数だけ条件を指定します。上から順番に条件式１、条件式２…と判定を行っていき、最初に条件式が正しいと判定されたところの処理を実行します。

曜日ごとに表示するメッセージを変更する

それでは先ほど書いた23行目からのif文を修正して、一週間分のメッセージを設定しましょう。波かっこがたくさん出てくるので、開始かっこと終了かっこの数が合っているかどうか気を付けながらタイピングして下さい。

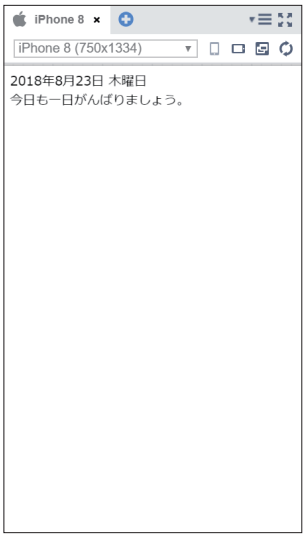
実習

サンプルプログラム

index.html

```
21      // 曜日を取得
22      let weekday = date.getDay();
23      if(weekday == 0) {
24          document.write("日曜日");
25      } else if(weekday == 1) {
26          document.write("月曜日");
27      } else if(weekday == 2) {
28          document.write("火曜日");
29      } else if(weekday == 3) {
30          document.write("水曜日");
31      } else if(weekday == 4) {
32          document.write("木曜日");
33      } else if(weekday == 5) {
34          document.write("金曜日");
35      } else if(weekday == 6) {
36          document.write("土曜日");
37      } else {
38          document.write("エラー！");
39      }
```

実行結果



今日の曜日に合わせたメッセージが表示されましたね。メッセージの内容は自分の生活に合わせてカスタマイズしてみても良いでしょう。後日、他の曜日にもアプリを起動して、メッセージが変わっていることを確認してみてください。

第 6 章

関数

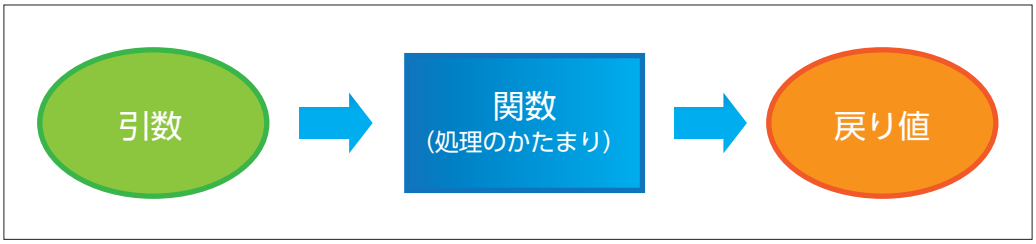
前章では日付と曜日を表示するアプリを作りましたが、コードを少しずつ追記してきたためにだんだんとプログラムが長くなってきました。それどころか本格的なアプリを作る場合は、コードが数百行、数千行、数万行といった途方もない長さになる場合もあります。そのような長いコードがずらずらと記述されていると非常に読みづらくなってしまうため、機能単位でコードを切り分けて、「関数」というひとまとまりを作り、それらを複数組み合わせることでプログラムを構築していきます。

クラシック・テンプレートを使用するか、サポートページから本章のひな形をインポートしてください。

関数とは

ある一連の処理をひとまとめにしたものを関数といいます。機能ごとに処理をまとめることでプログラムが読みやすくなり、何の処理がどこに記述されているかわかりやすくなります。また、よく使う処理を関数にしておけば、必要な場面で同じ処理を何度も実行することができるようになります。

関数は、処理をするために必要な情報である「引数」を受け取ります。そして、処理が終了すると「戻り値」と呼ばれる処理結果が返ってきます。



ただし、引数と戻り値は必須ではなく、省略することも可能です。

関数の書き方

まずは最も単純な、引数・戻り値ともに省略したパターンから見ていきましょう。

引数も戻り値もない関数

文法　関数の定義（引数・戻り値なし）

```
function 関数名() {
    処理;
}
```

文法　関数の呼び出し

```
関数名();
```

関数は、作っただけでは実行されません。まず関数として名前を付けたものを作っておいて、その関数を呼び出す命令を記述することではじめて関数内の処理が実行されます。つまり定義と呼び出しの、二段階の手順を踏まなければ関数は動かないということです。

実習

本章のひな形、もしくはクラシック・テンプレートを開き、index.html を編集しましょう。<script> タグ内に、以下のプログラムを記述します。

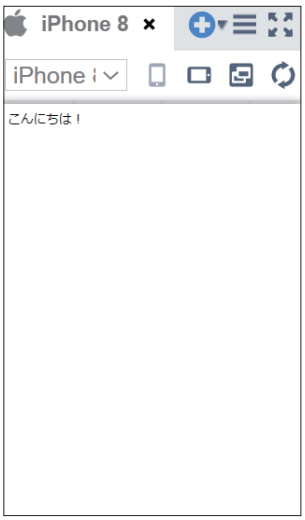
サンプルプログラム

index.html

```
10      <script>
11          // 関数を作る
12          function sayHello() {
13              document.write(" こんにちは！ ");
14          }
15
16          // 関数を実行
17          sayHello();
18      </script>
```

完成したら、プレビュー画面で実行結果を確認しましょう。関数が実行され、「こんにちは」という文字列が画面に出力されたことがわかります。

実行結果



17行目のsayHello関数を呼び出している処理が走ると、sayHello関数が定義されている12行目に飛んで、関数内に記述された処理が実行されます。

なお、関数を定義する処理は記述する順序に関係なく優先的に行われるので、以下のように関数の呼び出しが先に記述されていても同じように動作します。

解説

index.html

```
10      <script>
11          // 関数を実行
12          sayHello();
13
14          // 関数を作る
15          function sayHello() {
16              document.write("こんにちは！");
17          }
18      </script>
```

引数がある関数

引数とは、関数が処理をするために必要な情報のことです。例えば、ある数値を２倍する、という処理を行う関数があった場合、「ある数値」というのがいくつなのかを指定しなければなりません。この「ある数値」にあたる情報を引数と言います。

文法　関数の定義（引数あり）

```
function 関数名（引数を入れる変数名）{
    処理；
}
```

文法　関数の呼び出し

```
関数名（関数に渡す引数）;
```

関数は、引数を受け取るとそれを変数に代入します。その変数を宣言しているのがfuntion文の丸かっこ内です。このときは let をつける必要はありません。

また、関数は以下のようにカンマ区切りで複数指定することも可能です。

文法　関数の定義（複数の引数あり）

```
function 関数名（引数を入れる変数名1，引数を入れる変数名2 …）{
    処理；
}
```

文法　関数の呼び出し

```
関数名（関数に渡す引数1，関数に渡す引数2 …）;
```

実習

引数を受け取る関数を作って、実行してみましょう。
先ほど書いたプログラムを、以下のように変更します。「最中　太郎」と記述されているところには、自分の名前を書いてみましょう。

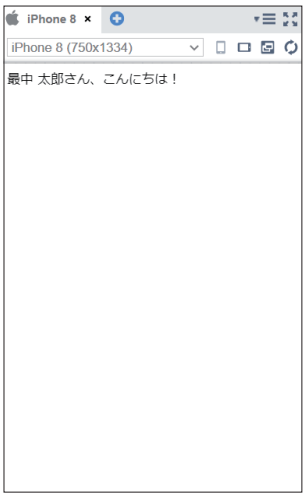
サンプルプログラム

index.html

```
10      <script>
11          // 関数を作る
12          function sayHello(name) {
13              document.write(name + " さん、こんにちは！ ");
14          }
15
16          // 関数を実行
17          sayHello(" 最中 太郎 ");
18      </script>
```

完成したら、プレビュー画面で実行結果を確認しましょう。最初に名前を呼んでから挨拶をしてくれるようになります。

実行結果



17行目でsayHello関数を呼び出すときに、名前データを引数として渡すと、sayHello関数は受け取った名前データを変数nameに代入し、それから関数内の処理を実行します。

戻り値がある関数

戻り値とは、関数の処理結果のことです。ある数値を２倍する関数があった場合、２倍した値を結果として返却しなければなりません。この結果を戻り値といいます。引数は呼び出したところから関数にデータを渡していましたが、戻り値はその逆で関数から呼び出したところにデータを渡します。

関数の定義（戻り値あり）

```
function 関数名() {
    処理;
    return 戻り値;
}
```

関数の呼び出し

```
let 戻り値を入れる変数名 = 関数名();
```

return という文が戻り値を呼び出し元に渡す命令です。呼び出し元は、戻り値を受け取ったらそれを変数に代入しなければならないので、イコール記号を使って変数へ値を代入する式を記述します。

実習

引数を受け取る関数を作って、実行してみましょう。

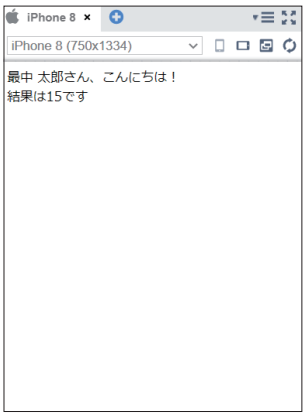
先ほど書いたプログラムの下の19行目以降に、以下のコードを追記します。

サンプルプログラム

index.html

```
19          // 戻り値を返す関数
20          function calc() {
21              let num = 1 + 2 + 3 + 4 + 5;
22              return num;
23          }
24
25          // 関数を実行し、戻り値を受け取る
26          let result = calc();
27          document.write("<br>");
28          document.write(" 結果は " + result + " です ");
29      </script>
```

実行結果



戻り値がある場合の流れは少し複雑なので、順を追って見ていきましょう。

まず、26 行目の calc 関数を呼び出す処理が実行されて、20 行目に飛びます。calc 関数内では、1 から 5 までの数値を加算した結果を求めて、変数 num に代入し、その値を戻り値として返却しています。するとまた 26 行目に戻ってきて、戻り値である 15 という値を変数 result に代入しています。
まとめると、戻り値のある関数を呼び出した場合は、以下のような流れでプログラムが動いていきます。

① 関数を呼び出す。
② 関数内の処理が実行される。
③ 関数が戻り値を返す。
④ 呼び出し元が戻り値を受け取る。

西暦を和暦に変換する

最後に応用問題として、引数・戻り値の両方があるプログラムを作成してみましょう。
西暦の年を引数として渡すと、和暦に変換して返却する関数を作ります。今回は令和年号へのみ変換可能な簡易的なプログラムを作成します。

実習

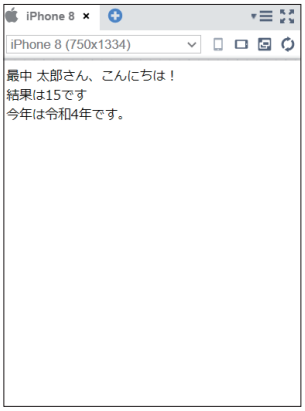
サンプルプログラム

index.html

```
30          // 西暦から和暦に変換する
31          function convertYear(western) {
32              let japanese = western - 2018;
33              return japanese;
34          }
35
36          // 今年が西暦何年かを取得する
37          let date = new Date();
38          let year = date.getFullYear();
39          // 和暦に変換
40          let japaneseYear = convertYear(year);
41          document.write("<br>");
42          document.write(" 今年は令和 " + japaneseYear + " 年です。");
43      </script>
```

実行すると、今年が令和何年かが表示されます。

実行結果



40 行目で convertYear 関数が呼び出されて、31 ～ 34 行目の convertYear 関数が実行されます。このとき、引数として西暦の年を渡しています。convertYear 関数内では、受け取った西暦から和暦を算出しています。令和元年が 2019 年なので、西暦から 2018 を引けば令和の年に変換することができます。このようにして変換した結果を戻り値として 40 行目の呼び出し元に返却しています。

関数は次章以降でも頻繁に利用しますので、使い方は複雑ですがここでしっかりと理解をしておいてください。

第 7 章

イベント

これまでに作ったアプリは起動時にしか処理が動いていなかったので、ユーザーは最初に表示された画面をただ見ることしかできませんでした。皆さんが普段使っているアプリには、「メニューアイコンが押されたらメニューを表示する」「ボタンが押されたら登録を行う」といった機能があると思います。このようにユーザーの操作を受けてから何らかの処理を行うには、「イベント」という仕組みを利用します。

実習の前にサポートページから本章のために用意されたひな形をインポートしてください。

イベントとは

ボタンをクリック／タッチした、画面をスワイプした、などのアプリ上で起こった出来事のことを「イベント」と呼びます。たとえば、「ユーザーが画面上の部品を押したら、警告メッセージを表示する」という動作を実現したい場合などに利用します。このイベントという仕組みを利用するには、あらかじめ画面上の部品を表す HTML 要素に、「このイベントが発生したら、この関数を実行する」といったように要素に対してイベントの関連付けを行っておきます。

文法　HTML 要素にイベントを関連付ける

```
<タグ名 onイベント名="関数名()">
```

例　ボタンがクリックされたときに test 関数を実行する

```
<button onclick="test()">ボタン</button>
```

イベントは、タグの属性として記述します。以下にアプリ制作の際によく使われるイベントをまとめています。パソコンで動く Web サイトであれば 左クリック／右クリック／ダブルクリック／マウスカーソルを乗せるといったイベントもあるのですが、モバイルアプリの場合にはマウス操作をすることがないので、ここでは割愛します。

イベント一覧

イベント名	概要
click	HTML 要素がクリック、またはタップされたタイミングで発生します。
load	最初にページが表示されたタイミングで発生します。 このイベントは <body> 要素に対して付与します。
touchstart	モバイル端末の画面に指が触れたタイミングで発生します。
touchmove	モバイル端末の画面に指を触れたまま上下左右に動かしている間、複数回発生します。
touchend	モバイル端末の画面から指を話したタイミングで発生します。

この中でも、特によく使われるのはclickイベントとloadイベントです。本章ではこの２つのイベントの使い方を学習します。

実習

はじめに、clickイベントの使い方について学びます。
本章のひな形プロジェクトを開き、index.htmlを編集しましょう。
まずはボタンが押されたときにメッセージを表示するclickMessage関数を作成します。
11 ～ 13行目に、以下の関数を追加してください。

サンプルプログラム

index.html

```
10      <script>
11          function clickMessage() {
12              alert(" ボタンが押されました ");
13          }
14      </script>
```

つづいて、「クリック／タップされたとき」に発生するclickイベントと、clickMessage関数を関連付けます。17行目の<button>タグに、onclick属性を追加してください。

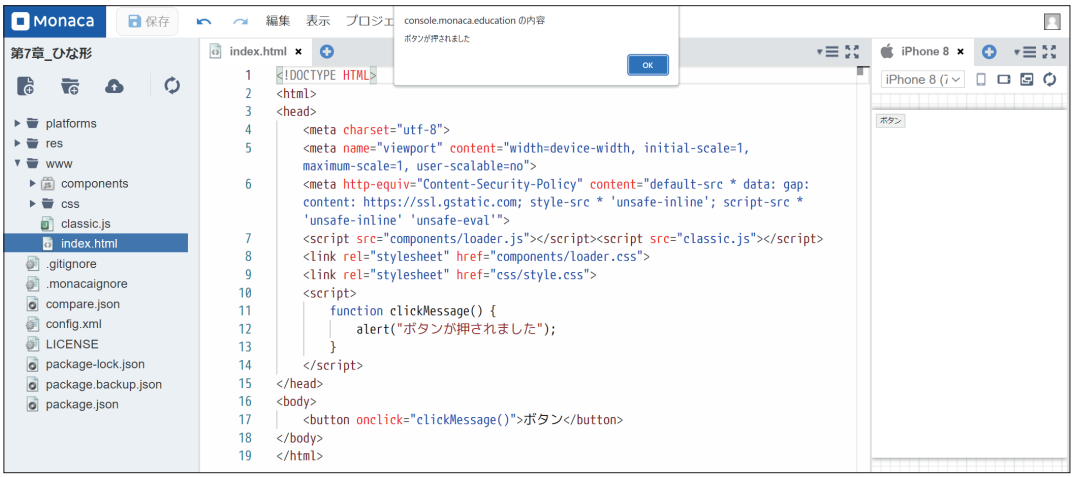
サンプルプログラム

index.html

```
17      <button onclick="clickMessage()"> ボタン </button>
```

ここまでできたら、保存してプレビュー画面を確認してみましょう。ボタンをクリックすると、以下のようにダイアログが表示されます。

実行結果



このように、以下の二段階の手順を踏むことでイベントを利用することができます。

1．イベント発生時に実行したい関数を作る。

2．関数とイベントを関連付ける。

実習

次に、ページが開いたときに発生する load イベントの使い方を学びます。index.html の 14 ～ 16 行目に、loadMessage 関数を作成してください。

サンプルプログラム

index.html

```
14          function loadMessage() {
15              alert("こんにちは");
16          }
```

つづいて、load イベントと、作成した loadMessage 関数を関連付けます。19 行目の <body> タグに、onload 属性を追加してください。

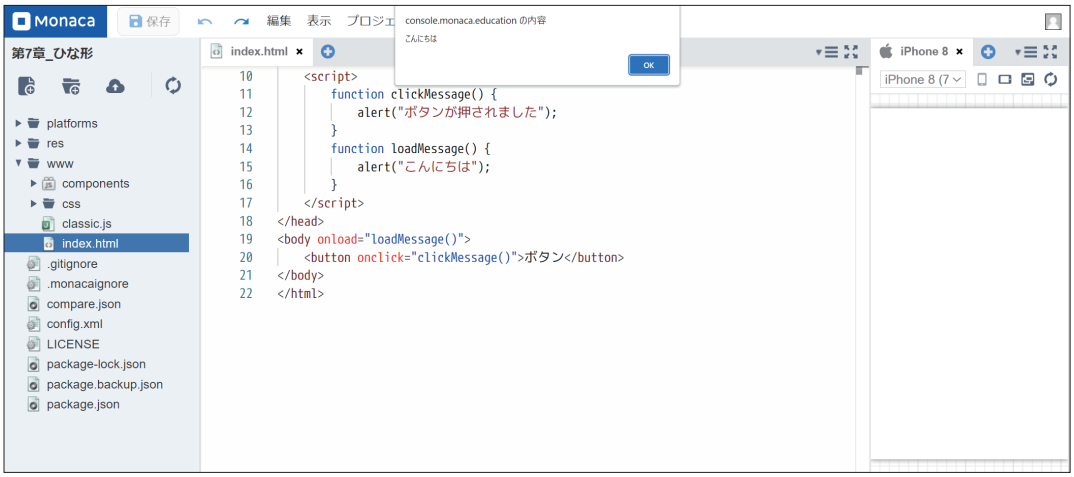
サンプルプログラム

index.html

```
19 <body onload="loadMessage()">
```

ここまでできたら、保存してプレビュー画面を確認してみましょう。

実行結果



loadイベントはページが開いてすぐに実行されるので、イベントを利用する意味がないように感じられるかもしれません。今回のようなダイアログを表示するだけの処理であれば、イベントを使わずに以下のように記述しても全く同じ結果を得ることができます。

```
<script>
    alert("こんにちは");
</script>
```

では、loadイベントはどのような場合で使うのかというと、次章で学ぶDOMを利用する際に必要となります。ひとまずここでは、最初に実行されるイベント、と覚えておいてください。

ポイント

イベントを扱う方法はタグの属性として記述するやり方の他にも「addEventListener」という命令を使うやり方もあります。HTML タグ側を編集せずにイベントを扱えるため必要に応じて使い分けます。

第8章

DOM

ここまで、JavaScript から画面に文字列を表示する方法として「document.write」という命令を使ってきました。しかしこの命令は実際のアプリ開発ではまず使うことはありません。一般的にアプリを作るときは HTML で表示内容を定義し、CSS でデザインを適用してから JavaScript を記述するため、「document.write()」で既に完成している画面の <body> タグ内に文字列を挿入すると、せっかく綺麗に作った画面が崩れてしまうのです。
そこで、すでに完成している画面に対して変更の操作を加えるための仕組みとして、DOM という技術があります。DOM を使うと、表示内容の変更はもちろん、タグの属性やスタイルの変更などがとても簡単に行えるようになります。

実習の前にサポートページから本章のために用意されたひな形をインポートしてください。

DOMとは

DOM（Document Object Model）は、HTMLで書かれた文書の各要素にアクセスするための仕組みです。アプリを起動して画面が読み込まれたタイミングでDOMという仕組みが働き、各要素を操作するための機能がJavaScriptで利用できるようになります。

DOMが利用可能になるタイミング

DOMはアプリ起動時に利用可能となるのですが、厳密にはHTMLファイル内のすべてのタグが読み込まれてからでないと利用することができません。読み込みにかかる時間は1秒以下なので人間が体感することはまずありませんが、DOMが利用可能になるよりも先に<script>タグ内に記述されたJavaScriptが実行されてしまうので、JavaScriptの書き方に工夫が必要になります。
そこで、前章で学んだloadイベントが役立ちます。このイベントは<body>タグの中身がすべて表示された後に発生するので、loadイベント発生時にはDOMも利用可能になっているというわけです。

ポイント

DOMが利用可能になった後にJavaScriptを実行する別の方法として、「プログレッシブ・テンプレート」を使う方法もあります。このテンプレートはloadイベントの記述をしなくてもDOMを操作できます。このテンプレートではJavaScriptを「main.jsファイル」に記述する方式を採用しているのですが、ファイルの読み込みがDOM構築後に行われるように調整されているため1行目からDOMを操作できます。

HTML要素へのアクセス

DOMの機能を使うと、JavaScriptでHTML要素を操作することができます。JavaScriptからHTML要素へアクセスするには、以下の命令を使います。

文法　要素へのアクセス

```
document.getElementById("ID値")
```

Elementは要素という意味ですので、getElementByIdという名前は「IDを指定してHTML要素を取得する」という意味になります。ここで呼んでいるIDというのは、タグに指定したID属性値のことです。ですから、この命令を使うためにはあらかじめ対象のHTML要素にID属性を付けておかなければなりません。

要素の内容を変更する

HTML 要素にアクセスすることが出来たら、続けて要素に対する操作を記述します。要素の内容（開始タグと終了タグで囲まれた文字列）を書き換える場合は、以下のように記述します。

文法　内容の変更

```
document.getElementById("ID値").innerHTML = "書き換える内容";
```

innerHTML というのは、要素の内容のことです。ここに文字列を代入することで、内容を上書きする処理が行われます。

実習

本章のひな形プロジェクトを開き、HTML と CSS の内容を確認してから、プレビュー画面を見てください。あらかじめ画面のデザインが作られていることがわかるはずです。

解説

index.html

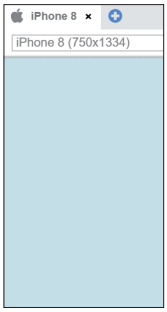
```
13 <body>
14     <p id="message"></p>
15     <img id="icon" src="">
16 </body>
```

解説

style.css

```
 1 body {
 2         background-color: #D0E7EF;
 3         text-align: center;
 4 }
 5 #message {
 6         margin-top: 60px;
 7         font-size: 26px;
 8         color: #367996;
 9 }
10
```

実行結果



プレビュー画面には水色の何もない画面が表示されます。ただし画面に表示はされていなくても、<body>タグ内にはHTML要素が定義されています。まずは14行目の 要素の開始タグと終了タグの間に時間帯に合わせたあいさつを挿入してみます。まず、<script>タグ内にgreet関数を定義します。

サンプルプログラム

index.html

```
10      <script>
11          function greet() {
12              // 現在の時刻を取得
13              let date = new Date();
14              let hour = date.getHours();
15
16              if(hour >= 5 && hour <= 10) {
17                  document.getElementById("message").innerHTML = "おはよう！";
18              } else if(hour >= 11 && hour <= 18) {
19                  document.getElementById("message").innerHTML = "こんにちは！";
20              } else {
21                  document.getElementById("message").innerHTML = "こんばんは！";
22              }
23          }
24      </script>
```

そして、このgreet関数をloadイベント発生時に実行するように<body>タグを修正します。

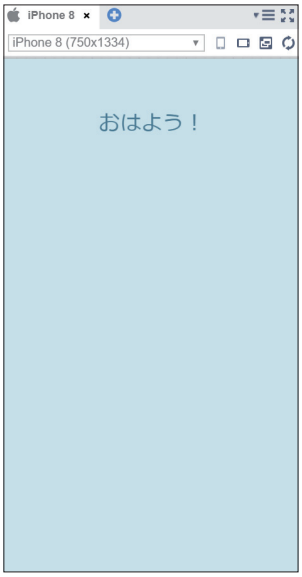
サンプルプログラム

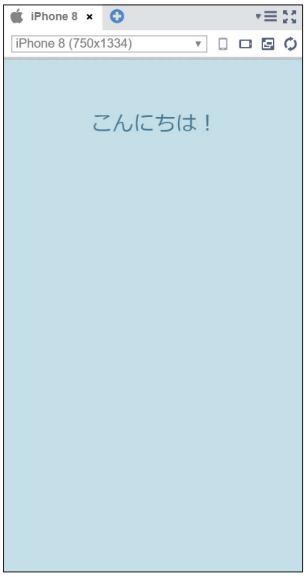
index.html

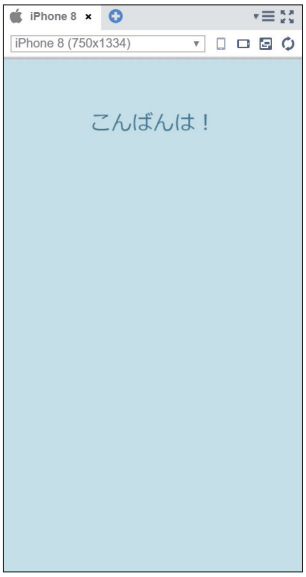
```
26  <body onload="greet()">
```

実行すると、時間帯に応じて異なるメッセージがアプリ内に表示されます。

実行結果







greet 関数内では、24 時間表記の現在の時間（0 ～ 23）を取得し、その値を元に条件分岐を行っています。

16 行目と 18 行目の条件式の中に、「&&」という見慣れない記号が出てきました。この記号には、2 つの条件式を連結する役割があります。左右に記述した条件式がどちらも正しかった場合にのみ、条件が成立します。

16 行目の条件式を例にとってみると、現在の時間が 5 時台から 10 時台の範囲だった場合にのみ「朝」の時間帯であると判断しています。

&&演算子の使い方

```
   hour >= 5    &&    hour <= 10
（時間が5以上）かつ（時間が10以下）
```

このように複数の条件式を組み合わせることで、より詳細な条件を指定することができるようになります。

そして、実際に画面にあいさつを表示しているのは17行目、19行目、21行目です。それぞれ、「message」というIDを持つ要素の内容に、あいさつの言葉を設定しています。仮に現在時刻が夜の範囲だった場合は、以下のように<p>要素が書き換えられています。

実行前

```
<p id="message"></p>
```

実行後

```
<p id="message">こんばんは</p>
```

要素の属性・スタイルを変更する

操作できるのはタグの内容だけではありません。タグの属性も JavaScript で変更することができます。innerHTML のかわりに属性名を記述し、そこに値を代入すると、既存の属性値が上書きされます。例えば <img> タグの src 属性を変更すれば、表示される画像を切り替えることができます。

文法　属性の変更

```
document.getElementById("ID値").属性名 = "属性値";
```

例　画像を「flower.jpg」に切り替える

```
document.getElementById("target").src = "flower.jpg";
```

また、以下のように記述することで要素のスタイルを変更することも可能です。

文法　スタイルの変更

```
document.getElementById("ID値").style.CSSプロパティ名 = "値";
```

例　要素の外側余白を 20px に設定する

```
document.getElementById("target").style.margin = "20px";
```

例外として、「background-color」などのハイフン (-) が含まれる CSS プロパティを変更する場合は特殊な記述方法になるので注意してください。プロパティ名にハイフンが含まれる場合は、ハイフンを除去して、ハイフンの後ろの英字を大文字に変更しなければならないという規則があります。

例　要素の背景色を青に設定する

```
document.getElementById("target").style.backgroundColor = "blue";
```

実習

先ほどのプログラムに手を加えて時間帯に応じた画像を表示してみましょう。<img>要素のsrc属性を変更することで、表示する画像を切り替えることができます。以下の赤枠で囲まれた行を追記してください。

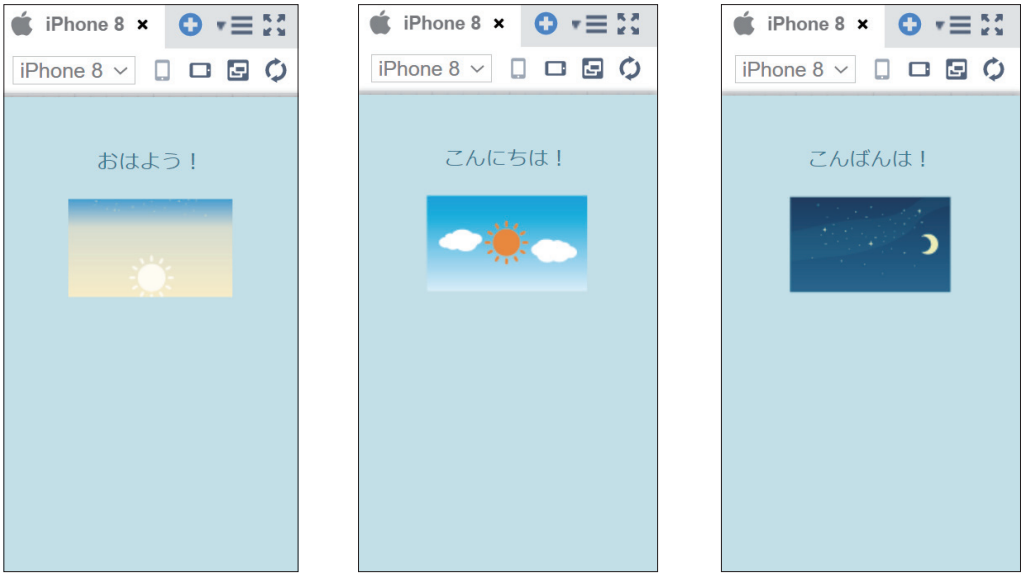
サンプルプログラム

index.html

```
16              if(hour >= 5 && hour <= 10) {
17                  document.getElementById("message").innerHTML = "おはよう！";
18                  document.getElementById("icon").src = "images/morning.png";
19              } else if(hour >= 11 && hour <= 18) {
20                  document.getElementById("message").innerHTML = "こんにちは！";
21                  document.getElementById("icon").src = "images/afternoon.png";
22              } else {
23                  document.getElementById("message").innerHTML = "こんばんは！";
24                  document.getElementById("icon").src = "images/evening.png";
25              }
```

時間帯に合わせた画像が表示されることを確認しましょう。

実行結果



最後に、スタイルを変更して時間帯が夜だった場合だけ背景色を暗くしてみましょう。25行目に背景色を変更する処理を追記してください。

サンプルプログラム

index.html

```
16              if(hour >= 5 && hour <= 10) {
17                  document.getElementById("message").innerHTML = "おはよう！";
18                  document.getElementById("icon").src = "images/morning.png";
19              } else if(hour >= 11 && hour <= 18) {
20                  document.getElementById("message").innerHTML = "こんにちは！";
21                  document.getElementById("icon").src = "images/afternoon.png";
22              } else {
23                  document.getElementById("message").innerHTML = "こんばんは！";
24                  document.getElementById("icon").src = "images/evening.png";
25                  document.body.style.backgroundColor = "black";
26              }
```

<body>タグは画面全体を表す特別なタグなので、IDを指定しなくても「document.body」と指定するだけで要素にアクセスすることができます。この命令が実行されると、CSSファイルに以下のように記述したのと同じ状態になります。

```
body {
        background-color: black;
}
```

実行結果を確認するにあたって、もし皆さんが実習をしている時間が朝や昼の時間帯であれば、14行目の現在時間を取得する処理に対して数値を加算し、無理やり夜の時間帯に変更してしまいましょう。

サンプルプログラム

index.html

```
14              let hour = date.getHours() + 10;
```

実行結果

背景を暗くすることでより一層夜らしい雰囲気になりましたね。

あらかじめデザインを組み上げた状態のHTMLに対してDOM操作を行うことで、見栄えの良いアプリを作ることができました。HTMLで表示内容の定義を行い、CSSでデザインの定義し、JavaScriptで状態に応じて変動する処理を記述する、といったようにそれぞれの役割に合わせてプログラムを記述することがアプリを作る上でのコツとなります。

第9章

フォーム

本章では、ユーザーからの情報を入力させる「フォーム」について学習します。みなさんが普段使っているアプリにも、ユーザーの情報を入力する登録画面などがあると思います。フォームには、名前やメールアドレスなどの文字列を入力するテキストボックスや複数の選択肢を表示するプルダウンメニューなど、さまざまなパーツが用意されています。

実習の前にサポートページから本章のために用意されたひな形をインポートしてください。

フォームとは

テキストボックス、チェックボックス、プルダウンメニューなどの、ユーザーに情報を入力させるための部品をまとめたものをフォームと呼びます。第2章では触れませんでしたが、フォームはHTMLタグを使って作成します。

テキストボックス

任意の文字列を入力することができるフォーム部品です。

文法　テキストボックス

```
<input type="text">
```

value属性値は、テキストボックスの入力値と等しくなります。

例　テキストボックスの記述例

```
<input type="text" value="テスト">
```

テスト

チェックボックス

「はい」または「いいえ」のどちらかの状態をチェック記号で表現するフォーム部品です。

文法　チェックボックス

```
<input type="checkbox">
```

checked属性を付与すると、チェックが入っている状態になります。checked属性がない場合は、未チェック状態となります。

例　チェックボックスの記述例

```
<input type="checkbox" checked>
```

ドロップダウンメニュー

複数の選択肢の中からいずれか1件を選択させるためのフォーム部品です。

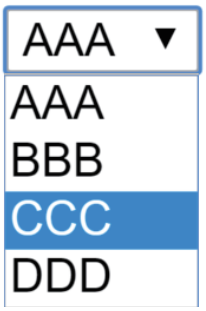
文法　ドロップダウンメニュー

```
<select>
        <option value="選択肢の値">表示する選択肢</option>
        <option value="選択肢の値">表示する選択肢</option>
        ⋮
</select>
```

選択肢の数だけ、<option>タグを記述します。value属性に指定した値は、画面上には表示されませんが、JavaScriptでユーザーが選択した値を取得すると、value属性の値が得られます。たとえば、以下の例で選択肢「CCC」がユーザーによって選ばれている場合、JavaScriptで選択されているものがどれであるかを調べると、「3」という値が取得されます。

例　ドロップダウンメニューの記述例

```
<select>
        <option value="1">AAA</option>
        <option value="2">BBB</option>
        <option value="3">CCC</option>
        <option value="4">DDD</option>
</select>
```



実習

本章のひな形プロジェクトを開き、index.html の <body> タグ内の内容とプレビュー画面を確認してください。ユーザー情報の登録画面を模したサンプルとなっています。

解説

index.html

```
13  <body>
14      <h1> フォームの練習 </h1>
15      <h2> 名前 </h2>
16      <input type="text" id="name">
17      <h2> 自動ログイン </h2>
18      <input type="checkbox" id="autoLogin" checked> 次回から自動ログインする
19      <h2> 言語設定 </h2>
20      <select id="languageList">
21          <option value="japanese"> 日本語 </option>
22          <option value="english"> 英語 </option>
23          <option value="chinese"> 中国語 </option>
24      </select>
25      <hr>
26      <button onclick="showProfile()"> 完了 </button>
27  </body>
```

16行目にテキストボックス、18行目にチェックボックス、20 ～ 24行目にドロップダウンメニューが定義されています。
これらのフォーム部品に入力された情報を JavaScript で取得するプログラムを作成します。フォームの入力情報を取得するには、前章で学んだ DOM という技術を使います。

<script> タグ内に、フォームの入力情報を表示する showProfile 関数を定義しましょう。

サンプルプログラム

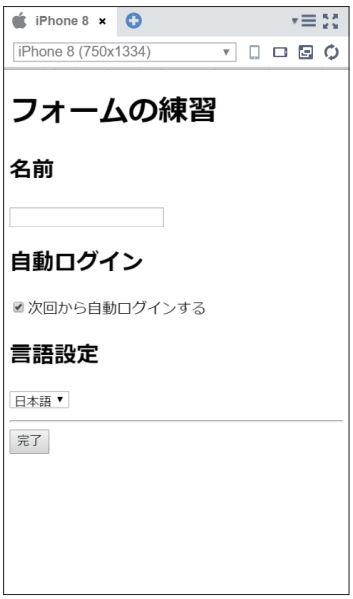
index.html

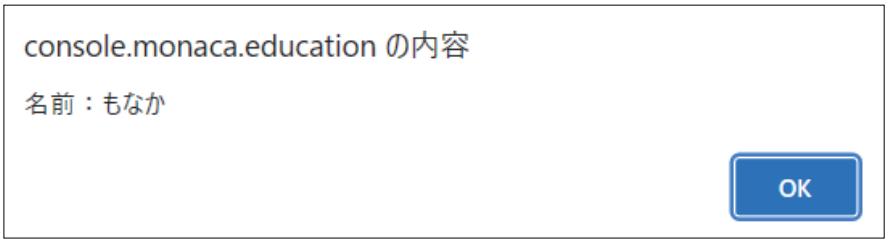
```
10      <script>
11          function showProfile() {
12              // テキストボックスの入力値を取得
13              let name = document.getElementById("name").value;
14              alert("名前：" + name);
15              // チェックボックスのチェック状態を取得
16              let checked = document.getElementById("autoLogin").checked;
17              if(checked) {
18                  alert("自動ログイン：ON");
19              } else {
20                  alert("自動ログイン：OFF");
21              }
22              // ドロップダウンメニューのうち選択されている値を取得
23              let language = document.getElementById("languageList").value;
24              alert("言語設定：" + language);
25          }
26      </script>
```

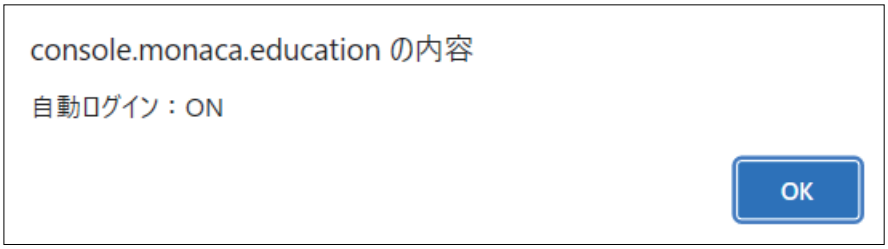
showProfile関数は、26行目の<button>タグをクリックしたときに実行されるようにイベント登録されています。

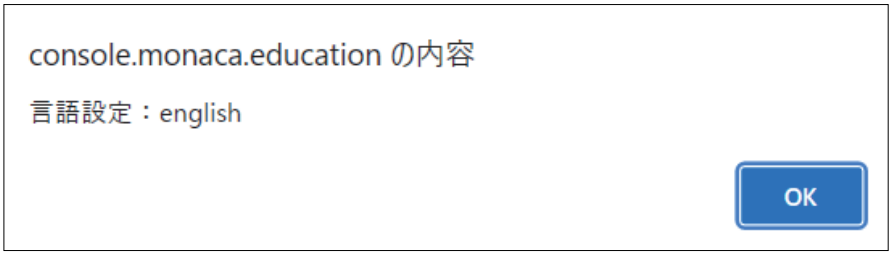
フォームに入力を行ったのち、「完了」ボタンを押して結果を確認してください。

実行結果









まず、IDを指定して各フォーム部品にアクセスします。テキストボックスの入力値と、ドロップダウンメニューの現在選択されている値は、どちらもvalue属性を指定することで取得することができます。チェックボックスのチェック状態を調べるには、checked属性を取得します。checked属性の値は、チェックが入っている状態であればtrue、チェックが入っていない状態であればfalseとなります。

10

第 10 章

いろいろな演算子

演算子とは、足し算を行う「+」や、比較を行う「==」などの、各種記号のことです。本章では、これまでに学んだ演算子以外の、さまざまな演算子を紹介します。

実習の前にサポートページから本章のために用意されたひな形をインポートしてください。

演算子の種類

四則演算子

四則演算子は、足し算や引き算などの算術演算を行うための演算子です。

演算子	概要	条件式の例	結果
+	数値の加算	1 + 1	2
+	文字列の連結	"Hello" + "World"	"HelloWorld"
-	数値の減算	2 - 1	1
*	数値の乗算	2 * 2	4
/	数値の除算	10 / 2	5

※　「+」演算子は、計算対象のデータ型に応じて処理内容が変わります。どちらか片方でも文字列であれば、結果は2つの値を連結した文字列となります。

算数の「＋」「－」「×」「÷」記号が、JavaScriptではそれぞれ「+」「-」「*」「/」に対応します。「加算・減算よりも乗算・除算が先に計算される」「丸かっこで括られた範囲を優先的に計算する」などの計算のルールは、JavaScriptも算数と同じです。

論理演算子

論理演算子は、主にif文の条件式を記述するために使用します。第五章では「今日が何曜日か」を調べるために比較演算子を使いましたが、「第何週目の何曜日か」といった、より複雑な条件を記述したい場合に論理演算子が必要になります。

演算子	概要	使用例	結果
&&	2つの条件式がどちらもtrueの場合はtrue	1 == 1 && 2 == 2	true
||	2つの条件式のどちらかがtrueの場合はtrue	1 == 2 || 2 == 2	true

論理演算子は、2つの条件式を結合します。「&&」演算子はAND演算子とも呼ばれ、2つの条件式がどちらもtrueだった場合に、結果としてtrueを返します。「||」演算子はOR演算子とも呼ばれ、2つの条件式のどちらか一方でもtrueであれば、結果はtrueになります。

ポイント　「|」記号について

「|」はパイプ、またはバーティカルバーと呼ばれる記号です。一般的な日本語キーボードの場合、「¥」記号と同じ位置にキーが配置されています。

複合代入演算子

複合代入演算子は、変数に対して算術演算と代入を同時に行う演算子です。

演算子	概要	使用例	結果（a の値）
+=	左辺の値に右辺の値を加算したものを代入	a = 1; a += 2;	3
-=	左辺の値から右辺の値を減算したものを代入	a = 5; a -= 2;	3
*=	左辺の値に右辺の値を乗算したものを代入	a = 3; a *= 2;	6
/=	左辺の値を右辺の値で除算したものを代入	a = 10; a /= 2;	5
++	変数に 1 加算する（インクリメント）	a = 1; a++;	2
--	変数から 1 減算する（デクリメント）	a = 1; a--;	0

たとえば、「a += 2」と記述した場合は、「a の現在の値に 2 を加算して、a を上書きする」という意味になります。現在変数に代入されている値を更新したい場合に利用します。

なお、「+=」記号は「+」記号と同様に、数値が演算対象の場合は足し算、文字列が演算対象の場合は文字列の連結となります。

BMI の求め方

四則演算子と論理演算子を使って、BMI 計算アプリを作成してみましょう。BMI とは、身長と体重の相関関係から求められる、体格指数のことです。痩せ・標準体重・肥満などの判定を行うために用いられる指標です。BMI は、以下の計算式で求めることができます。

BMI の計算式

```
体重 ÷（身長²）
```

※体重の単位は kg、身長の単位は m（cm ではないことに注意）

例　身長 170 ｃｍ、体重 60kg の人の場合

```
60 ÷ (1.7²) = 20.761...
```

BMI 判定表

状態	指標
低体重（痩せ型）	18.5 未満
普通体重	18.5 以上、25 未満
肥満（1 度）	25 以上、30 未満
肥満（2 度）	30 以上、35 未満
肥満（3 度）	35 以上、40 未満
肥満（4 度）	40 以上

実習

本章のひな形プロジェクトを開き、index.html の <body> タグ内の内容とプレビュー画面を確認してください。

解説

index.html

```
13 <body>
14     <h1>BMI 計算 </h1>
15     <h2> 身長 </h2>
16     <input type="text" value="" id="height">cm
17     <h2> 体重 </h2>
18     <input type="text" value="" id="weight">kg
19     <hr>
20     <button onclick="calc()"> 計算 </button>
21     <p id="bmi"></p>
22     <p id="message"></p>
23 </body>
```

入力フォームはすでに完成していますので、20 行目の「計算」ボタンを押したときに実行される処理を JavaScript で記述していきます。

サンプルプログラム

index.html

```
10      <script>
11          function calc() {
12              //  身長を取得
13              let height = document.getElementById("height").value;
14              // cm から m に変換
15              height = height / 100;
16              // 体重を取得
17              let weight = document.getElementById("weight").value;
18              // BMI を求める
19              let bmi = weight / (height * height);
20              document.getElementById("bmi").innerHTML = "BMI : " + bmi;
21
22              let message = " あなたは ";
23              if(bmi < 18.5) {
24                  message += " 低体重 ( 痩せ型 ) です。";
25              } else if (bmi >= 18.5 && bmi < 25) {
26                  message += " 普通体重です。";
27              } else if (bmi >= 25 && bmi < 30) {
28                  message += " 肥満 (1 度 ) です。";
29              } else if (bmi >= 30 && bmi < 35) {
30                  message += " 肥満 (2 度 ) です。";
31              } else if (bmi >= 35 && bmi < 40) {
32                  message += " 肥満 (3 度 ) です。";
33              } else if (bmi >= 40) {
34                  message += " 肥満 (4 度 ) です。";
35              }
36              document.getElementById("message").innerHTML = message;
37          }
38      </script>
```

実行結果

まずはじめにユーザーによって入力された身長と体重を取得しています。フォームから入力される身長はcm単位なので、BMIを求めるにはあらかじめcmからmに変換しておかなければなりません。15行目で身長を100で割ることで単位変換を行っています。

つづいて19行目で計算式に基づいてBMIを求め、23 ～ 35行目でBMI値を元に肥満度の判定処理を行っています。肥満度は、BMIの数値が指定された範囲内であるかどうかという条件で判定を行っています。たとえば「肥満（1度）」であるかどうかの判定条件を文章で正確に表現すると、「BMIが25以上でかつ、BMIが30未満である」といったように、2つの条件の組み合わせであることがわかります。両方の条件式を満たす必要があるので、この場合はAND演算子を利用することになります。

また、変数messageには初期値として「あなたは」という文字列が代入されています。そこに、if文の中で複合代入演算子を用いて文字列を追加しています。結果、「あなたは普通体重です」といったように文字列の連結が行われます。

このように、四則演算子や論理演算子を活用すると、より複雑な計算や判定を行うことができるようになります。たとえば税抜金額を入力して消費税を求めたり、預金額を入力して利息を求めたりといった、身近なところで役に立つアプリを作成することができます。今回のサンプルアプリを発展させて、どのようなアプリを作成することができるか考えてみると良いでしょう。

第 11 章

配列

これまで、数値や文字列などのデータは変数という入れ物の中に格納していました。扱うデータの件数が少ない場合は問題ありませんが、10個、20個、といったたくさんのデータを扱いたい場合、一つ一つ変数を宣言して用意するのは面倒です。そこで、たくさんのデータをまとめて格納することができる「配列」という仕組みが用意されています。

実習の前にサポートページから本章のために用意されたひな形をインポートしてください。

配列

配列とは、同じ用途で使う複数データを一括で管理するための仕組みです。変数の中に仕切りがあって、それぞれの部屋の中にデータを格納するイメージです。この部屋のことを「要素」と呼びます。たとえば 100 人分のテストの点数をデータとして扱う場合は、100 個分の要素を持つ配列を 1 つ作成します。

文法　配列の作成

```
let 配列名 = [要素1, 要素2, 要素3,  … ];
```

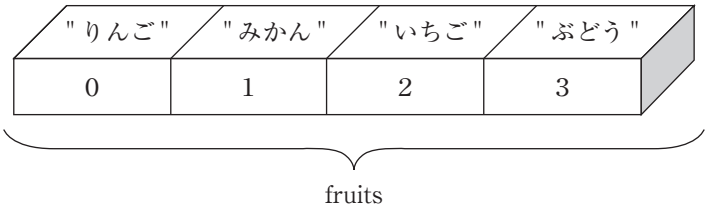
配列を作成する際は、角かっこの中にカンマ区切りで複数の要素を指定します。要素の数に制限はありません。また、変数と同じように let をつけて宣言することも忘れてはいけません。

例　配列の記述例

```
let fruits = ["りんご", "みかん", "いちご", "ぶどう" ];
```

上記のように記述した場合、要素 4 件分の配列が作成されます。

配列のイメージ



fruits という大きな変数の中に、4 つの要素が含まれているイメージです。1 つ 1 つの要素を識別するために、要素にはインデックスと呼ばれる 0 から始まる連番が割り振られます。

次に、配列内の各要素を参照し、格納されたデータを取得する方法を解説します。

文法　配列内の要素を参照する

```
配列名 [ インデックス ]
```

例　配列内の要素を参照する記述例

```
let fruits = ["りんご", "みかん", "いちご", "ぶどう"];
alert(fruits[2]);
```

上記のプログラムを実行すると、インデックスが２番目の要素に入っている値、「いちご」が表示されます。インデックスの先頭は１ではなく０から始まることに注意してください。

配列を利用することで、冗長なプログラムをすっきりと記述することができるようになります。第５章でif文を使って今日の曜日を判定するプログラムを作成しましたが、あのプログラムは実はif文を使わなくとも配列だけで作成することができます。if文は複雑な条件分岐を行う際に有効な文法であって、今日の曜日を判定するだけの単純なプログラムであれば、配列を使ったほうが適切であるといえます。

解説　if文による曜日判定

index.html

```
22          let weekday = date.getDay();
23          if(weekday == 0) {
24              document.write("日曜日");
25          } else if(weekday == 1) {
26              document.write("月曜日");
27          } else if(weekday == 2) {
28              document.write("火曜日");
29          } else if(weekday == 3) {
30              document.write("水曜日");
31          } else if(weekday == 4) {
32              document.write("木曜日");
33          } else if(weekday == 5) {
34              document.write("金曜日");
35          } else if(weekday == 6) {
36              document.write("土曜日");
37          } else {
38              document.write("エラー！");
39          }
```

解説　配列による曜日判定

index.html

```
22          let weekday = date.getDay();
23          let list = ["日曜日", "月曜日", "火曜日", "水曜日", "木曜日", "金曜日", "土曜日"];
24          document.write(list[weekday]);
```

最後に、配列の要素数を取得する方法を解説します。本章の実習では利用しませんが、次章の繰り返しで配列の要素数の取得が必要となります。

文法　配列の要素数を取得する

```
配列名.length
```

例　配列の要素数を取得する記述例

```
let fruits = ["りんご", "みかん", "いちご", "ぶどう"];
alert(fruits.length);
```

上記のプログラムを実行すると、要素数の「4」が表示されます。

実習

本章のひな形プロジェクトを開き、index.html の <body> タグ内の内容とプレビュー画面を確認してください。

解説

index.html

```
13  <body>
14      <h1> 心理テスト </h1>
15      <p> あなたが森の中を一人で散歩していると、ある生き物に出会いました。その生き物はどれですか？ </p>
16      <select id="choiceList">
17          <option value="0"> 鳥 </option>
18          <option value="1"> リス </option>
```

```
19          <option value="2"> クマ </option>
20          <option value="3"> 人間 </option>
21          <option value="4"> 妖精 </option>
22      </select>
23      <button onclick="showAnswer()"> 回答する </button>
24      <hr>
25      <p id="answer"></p>
26  </body>
```

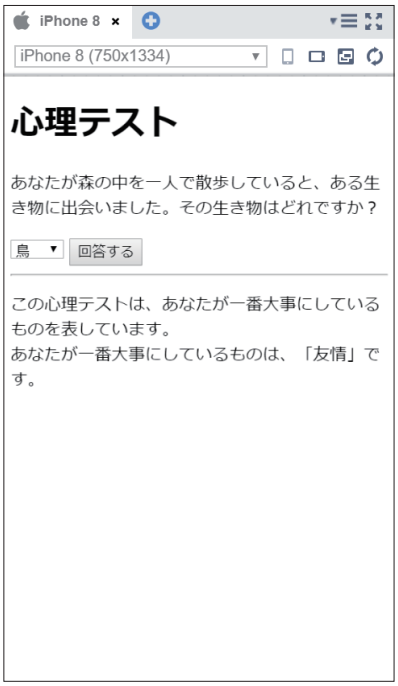
ユーザーに心理テストの回答をプルダウンメニューの中から選択させ、それに対応した結果を表示するプログラムを作成します。16 ～ 22 行目で定義しているプルダウンメニューが、心理テストの選択肢になります。ユーザーが選んだ選択肢に応じて、<option> タグの value 属性値である 0 ～ 4 のいずれかの値が JavaScript に渡されることになります。

サンプルプログラム

index.html

```
10      <script>
11          let answerList = [
12              " 友情 ",
13              " 愛 ",
14              " プライド ",
15              " お金 ",
16              " 夢 "
17          ];
18
19          function showAnswer() {
20              let choiceNo = document.getElementById("choiceList").value;
21              document.getElementById("answer").innerHTML = "";
22              document.getElementById("answer").innerHTML += " この心理テストは、
    あなたが一番大事にしているものを表しています。<br>";
23              document.getElementById("answer").innerHTML += " あなたが一番大事に
    しているものは、「" + answerList[choiceNo] + "」です。";
24          }
25      </script>
```

実行結果



11 ～ 17 行目では、あらかじめ心理テストの結果を配列 answerList として定義しています。先頭から順番にインデックスが割り振られますので、「友情」が 0 番、「愛」が 1 番、最後の「夢」は 4 番になります。

19 行目の showAnswer 関数は、「回答する」ボタンを押したときに実行される関数です。この関数の中で、心理テストの結果を の開始タグと終了タグの間に挿入しています。

21 行目では、""（ダブルクォート 2 つ）を p 要素の innerHTML に代入しています。ダブルクォート 2 つは空文字といい、その名のとおり空っぽの文字列であることを表しています。「回答する」ボタンを二回押したときに、一回目に表示された結果メッセージを消して、空っぽの初期状態に戻したいので、この処理が必要になります。

最後に 22、23 行目で結果メッセージを挿入しています。answerList[choiceNo] が、ユーザーが選んだ回答に対する結果になります。仮にユーザーが「鳥」という回答を選択した場合、choiceNo には <option> タグの value 属性値である「0」が代入されています。つまり answerList[0] となり、配列の 0 番目の要素である「友情」が表示されるという仕組みです。

配列を使うと扱うことのできるデータの幅が増えることがわかったかと思います。大量の情報を扱う場合は、変数を個別に用意するのではなく配列を利用すると覚えておきましょう。また、配列を扱う際は、次章で学ぶfor文と組み合わせる方法がよく用いられます。配列とfor文を組み合わせると、要素1つ1つに対して順番に処理を実行することができるようになります。

12

第 12 章

繰り返し

プログラムの作成中に、同じ処理を何回も繰り返し実行したい、という場面が出てくるはずです。例えば 1 ～100 までの数の合計を求める処理や、100個分の要素を持つ配列のデータを 1 件ずつ表示する処理などが挙げられるでしょう。同じ処理を何度も実行したい場合、コピー＆ペーストで処理を増やす方法ではキリがないので、繰り返しを行うための特別な文法を使って記述します。

実習の前にサポートページから本章のために用意されたひな形をインポートしてください。

for 文

ある処理を繰り返し実行するには、for 文という命令を利用します。for 文は、処理を繰り返し実行する中で、1 回繰り返すたびに変数の値が移り変わって行くという特徴があります。この値が移り変わって行く変数のことを、「カウンタ変数」と呼びます。

文法　for 文

```
for（初期化式；継続条件式；増減式）{
    繰り返す処理
}
```

for 文の記述方法は少々複雑です。丸かっこの中に、セミコロン（;）で区切って 3 つの式を記述します。それぞれの式の意味は以下の通りです。

式の種類	概要	実行するタイミング
初期化式	カウンタ変数を初期化する代入式	最初の 1 回だけ実行される
継続条件式	繰り返しを継続する条件式	1 回分の繰り返し処理の最初に毎回実行される
増減式	カウンタ変数の値を増減させる計算式	1 回分の繰り返し処理の最後に毎回実行される

少し難しい内容になりますので、実習を通して覚えていきましょう。

実習

「繰り返しの練習」プロジェクトを開き、index.html に以下の処理を追加してください。

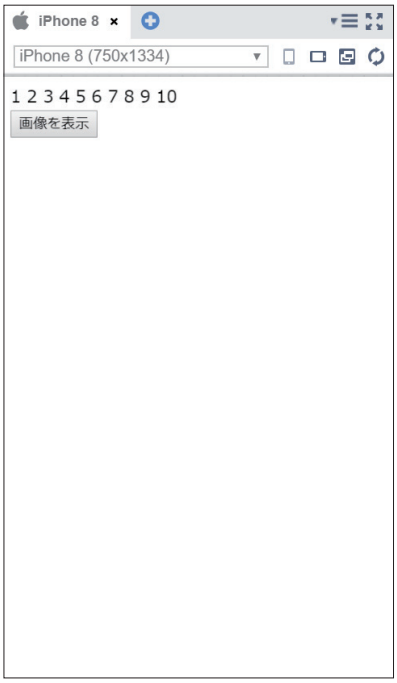
サンプルプログラム

index.html

```
10      <script>
11          for (let i = 1; i <= 10; i++) {
12              document.write(i);
13          }
14      </script>
```

実行結果

アプリを起動すると、以下のようにボタンの上に 1 から 10 までの数字が表示されます。




```
                    ②      ④
for (let i = 1; i <= 10; i++) {
        document.write(i);  ③
}
```

この for 文は、以下の順序で実行されます。

① カウンタ変数 i に 1 が代入される（初期化式）。
② i が 10 以下かどうか検証される（継続条件式）。
③ i の値が画面に表示される。
④ i に 1 加算される（増減式）。
※継続条件式の結果が false になるまで②～④を繰り返す。

したがって、計 10 回繰り返したことになります。for 文は、3 つの式をどのように指定するかによって繰り返し回数がかわります。カウンタ変数がいくつから始まって（初期化式）、いくつになったら終わって（継続条件式）、いくつずつ増えていくのか（増減式）の 3 つを指定する、ということを理解してください。

ポイント

カウンタ変数の名前は通常の変数と同様に自由ですが、慣例として i という名前がよく使われます。

for 文は配列を扱う際にも重宝します。配列のインデックスをカウンタ変数で指定することで、配列の各要素に対して 1 件ずつ順に処理をすることができます。

実習

index.html の 15 行目以降に、ボタンを押したときに実行される insert 関数を追記してください。

サンプルプログラム

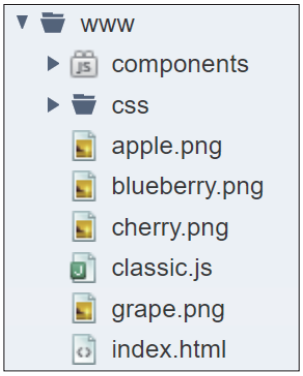
index.html

```
10      <script>
11          for (let i = 1; i <= 10; i++) {
12              document.write(i);
13          }
14
15          function insert() {
16              let images = [
17                  "strawberry.png",
18                  "orange.png",
19                  "grape.png",
20                  "peach.png",
21                  "melon.png"
22              ];
23
24              for (let i = 0; i < images.length; i++) {
25                  document.getElementById("imageList").innerHTML += "<img src=" +
    images[i] + ">";
26              }
27          }
28      </script>
```

実行結果

ボタンをクリックすると、画像が５つ表示されます。

16 ～ 22行目では、表示する画像ファイルを配列 images として定義しています。画像ファイルは [www] フォルダ以下に配置されています。



24 ～ 26行目で、for 文を使って の開始タグと終了タグの中に <img> タグを挿入する処理を５回繰り返しています。

for 文の 3 つの式を順に見ていくと、i は 0 から始まり、「images.length」未満の間繰り返し、1 つずつ増えていく、と指定されています。images.length は、配列 images の要素数です。5 つの画像ファイルを要素として持っているので、「5」を表します。つまり、i の値は 0,1,2,3,4 というように変化していきます。0,1,2,3,4 という値は、配列 images のインデックスと等しいので、ちょうどカウンタ変数 i の値が配列のインデックスと対応することになります。
この for 文が実行されると、以下のように HTML タグを記載したのと同じ状態になります。

```
<div id="imageList">
    <img src="strawberry.png">
    <img src="orange.png">
    <img src="grape.png">
    <img src="peach.png">
    <img src="melon.png">
</div>
```

繰り返しを使えば、数値でも文字列でも画像でも、大量のデータを扱う処理をほんの数行の記述で済ませることができます。

本書サポートページおよび正誤表は以下の URL からアクセスできます。

https://edu.monaca.io/template/

Monaca で学ぶアプリ制作入門 ～ HTMLxCSSxJavaScript 編～

2024 年 4 月 1 日 初版第 3 刷発行

著者　アシアル株式会社（アシアル情報教育研究所）
協力　株式会社 IMAKE
発行　アシアル株式会社
〒 113-0034
東京都文京区湯島２丁目３１－１４
ファーストジェネシスビル
https://edu.monaca.io/