Monacaで学ぶアプリ制作入門

～HTML×CSS×JavaScript編～

体験版

アシアル株式会社

Contents　目次

体験版につき4章まで収録

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、印刷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。例として、各章で学ぶ実習のひな形となるサンプルプロジェクトが掲載されています。

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

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まず公式サイトにアクセスしてアカウントを取得します。はじめてのプログラミングでは教育版、ビジネス利用では通常版がお勧めです。

教育版

通常版

Monaca のアカウント作成

教育版公式サイト URL にアクセスします。

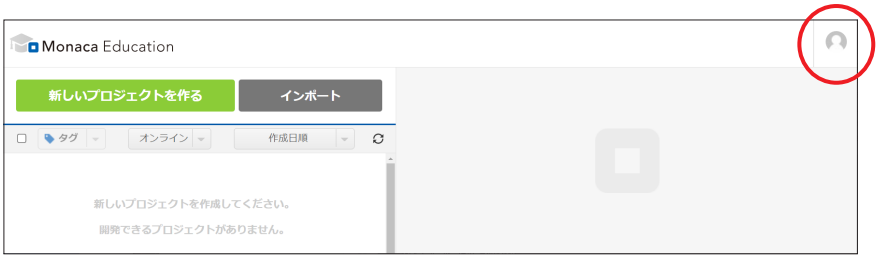
https://edu.monaca.io/

次に右上の「アカウント作成」をクリックして下さい。アカウント作成フォームが表示されますので、先生の指示に従ってアカウントを作成します。特に、Google アカウントや Microsoft アカウントと連携してアカウントを作る場合は注意して下さい。連携の場合、メールアドレスの入力は不要です。

メールアドレスの場合、登録した段階で仮登録状態となり本登録のためのメールが届きます。

メールに記載された URL にアクセスすることで登録が完了します。

本登録が完了すると 10 秒後にダッシュボードに移動します。

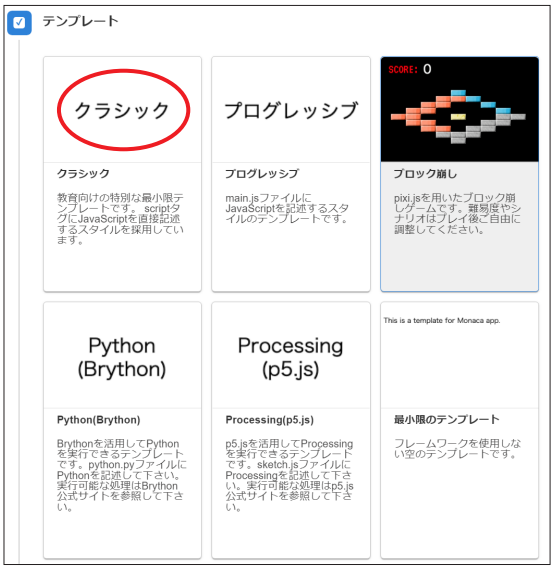


この画面では開発中のアプリをプロジェクトという単位で管理します。画面の左側にプロジェクトの一覧が表示されます。
なお無料プランの場合、保持できるプロジェクトは 3 個までとなっています。4 個目のプロジェクトを作成するためには古いプロジェクトを消すか有料プランに切り替える必要があります。有料プランを利用するためのアクティベーションコードをお持ちの場合は、「プラン管理」から適応できます。プラン管理は画面右上のアイコンからアクセスできます。
以上で Monaca のアカウント登録は完了です。

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

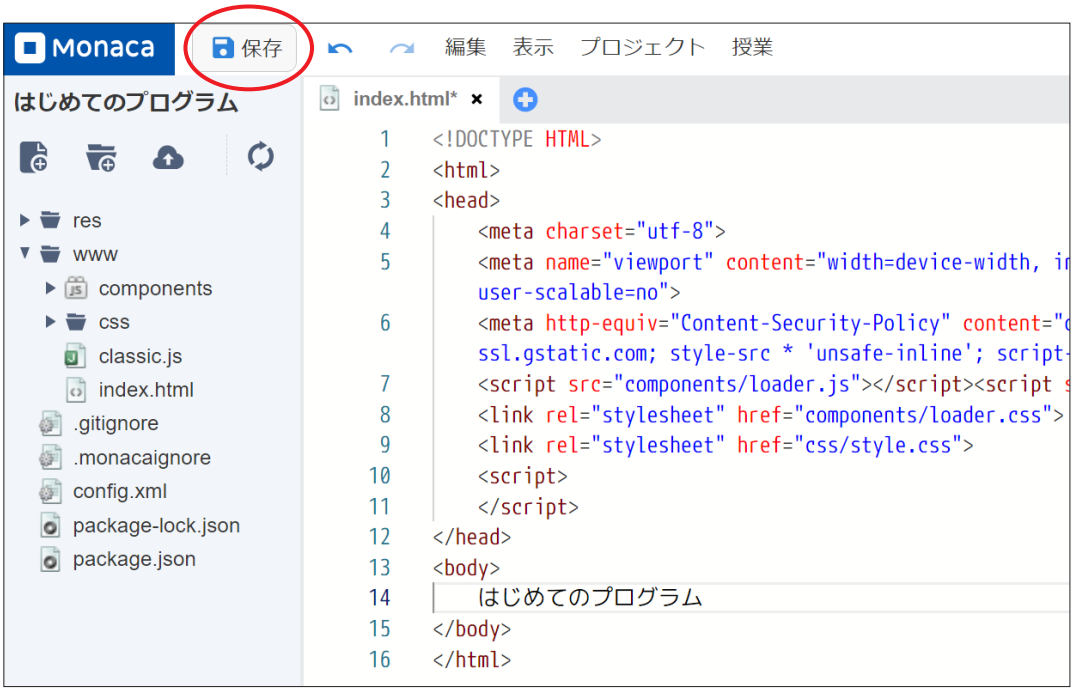
「はじめてのプログラム」をクリックし［クラウド IDE で開く］ボタンをクリックします。

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。

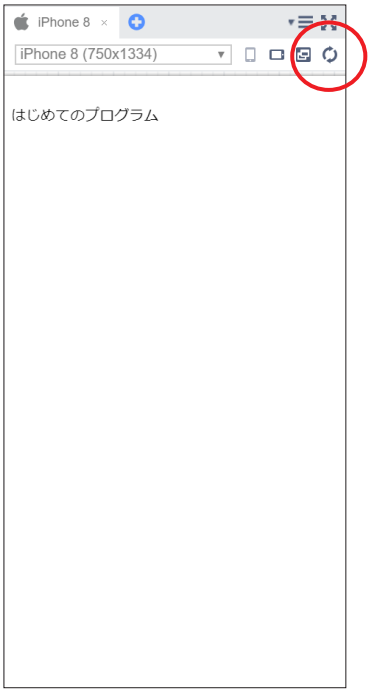
上部のパネルが「メニューバー」になります。プログラムを保存する機能やダウンロードする機能などはメニューバーから利用できます。

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

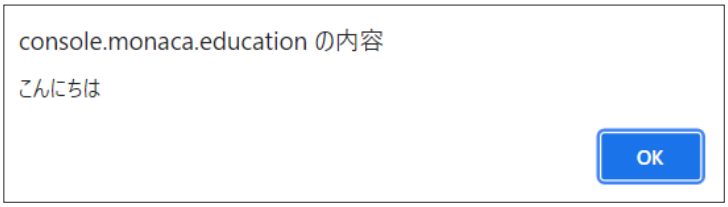
ダイアログとは、画面の前面に表示されるウィンドウのことです。ユーザーにメッセージを伝えたり、ユーザーから OK またはキャンセルといった操作を促したりするために使われます。<script> と </script> の間に alert("こんにちは"); という記述を行って保存し、プレビュー画面で確認してみてください。

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

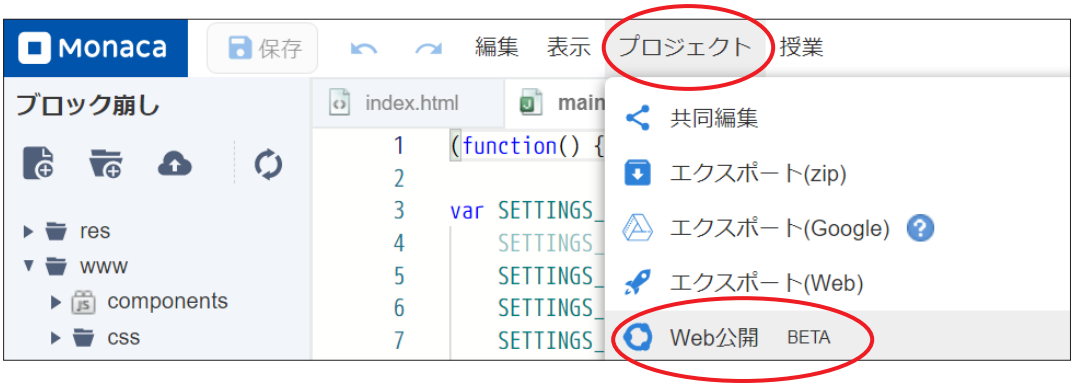


画面を開いたときに、ダイアログが表示されるようになりました。

Web 公開機能の利用

プレビューはMonacaにログインしていないと確認できないため、自分のスマートフォンでプロジェクトを動かしたり先生に提出したりすることができません。Web公開機能を使うことで、Web上にプロジェクトを作品として公開し、URLさえ知っていれば誰でもプレビューを行えるようにできます。

メニューの[プロジェクト]からWeb公開を選択して下さい。



選択すると公開の「On / Off」が選択できます。公開をOn にして右下のボタンを押下することで作品を公開できます。公開中の作品はログインしていなくても公開用のURL にアクセスすることで閲覧できます。また、QR コードも表示されるため、スマートフォンやタブレットから読み取って動かすこともできます。

(本機能の紹介はBETA 版の仕様に基づいています、2022 年度以降、一部の機能が変更される可能性があります。最新の仕様はサポートページを参照して下さい。)

Monaca for Study(デバッガーアプリ) の利用

先ほど利用した IDE のプレビュー機能や Web 公開機能はウェブブラウザで実行結果を確認する仕組みになっています。そのため、スマートフォンのカメラやコンパスなどのハードウェア機能の利用に制限があります。アプリとしてビルドする方法もありますが、ビルドには様々なコストが掛かるため学習目的の場合は「Monaca for Sutdy」という専用アプリの利用をお勧めしています。本アプリを使うことでプロジェクトを、スマートフォンやタブレットのアプリとして動かせます。なお通常版の Monaca では同様のアプリが「Monaca デバッガー」の名称で公開されていますので、注意してください。教育版では「Monaca for Study」を利用します。App Store または Google Play で「monaca」というキーワードで検索し、対応するデバッガーをインストールしてください。なお、本書の 1 ～ 12 章を学ぶだけであれば Monaca for Study の利用は必須ではありません。

教育版 (Monaca for Study)

通常版 (Monaca デバッガー)

インストールが終了したらアプリを起動して、先ほど Monaca に登録するときに使ったメールアドレスとパスワードを使ってログインしてください。ログインすると開発中のプロジェクト一覧が表示されます。動作確認したいプロジェクトの名前をタップすると、実行結果が表示されます。

なお、アカウント連携機能で Monaca に登録した場合は、パスワードが存在しないため IDE のメニュー [授業] にある「ワンタイムパスワード」機能でログインします。また、IDE の表示設定を変更することで、Monaca for Study アプリ内で発生したログやエラー情報を IDE 上に表示することもできます。詳しくはサポートページを確認してください。

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。

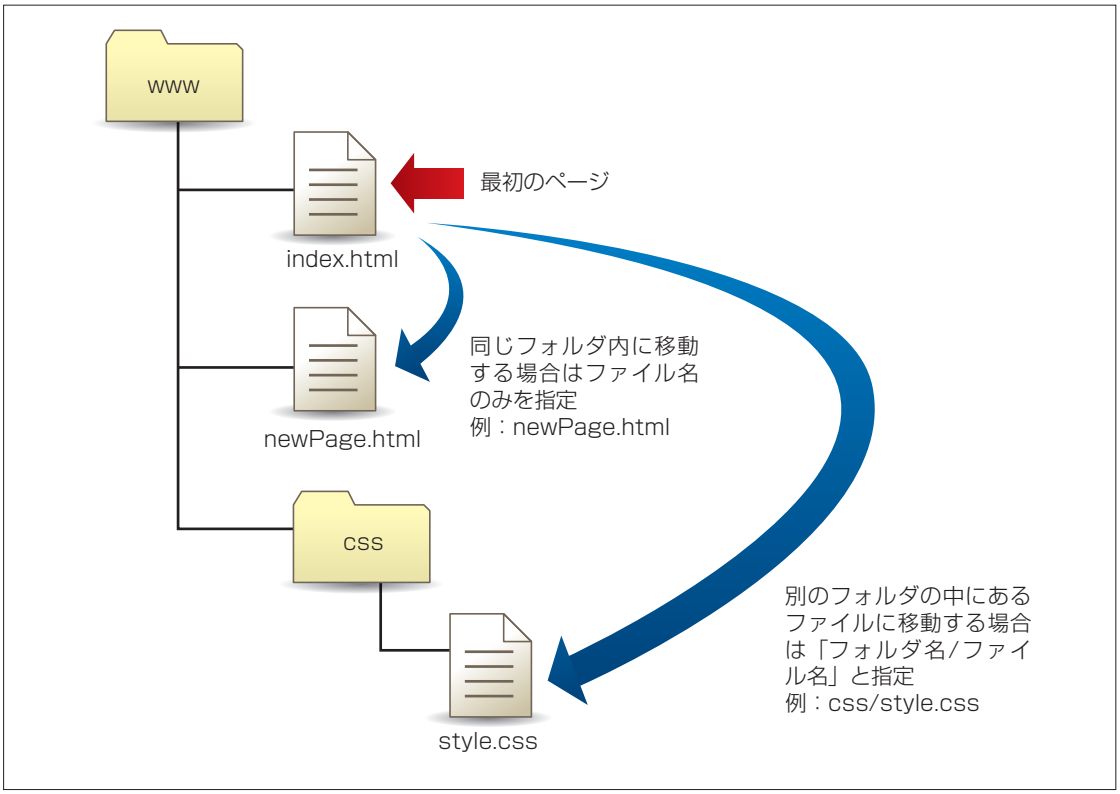
なお、リンク先によってはプレビュー機能では確認できない場合があります。その場合は、「Web 公開機能」や「Monaca for Study」アプリを使って確認してみましょう。

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。

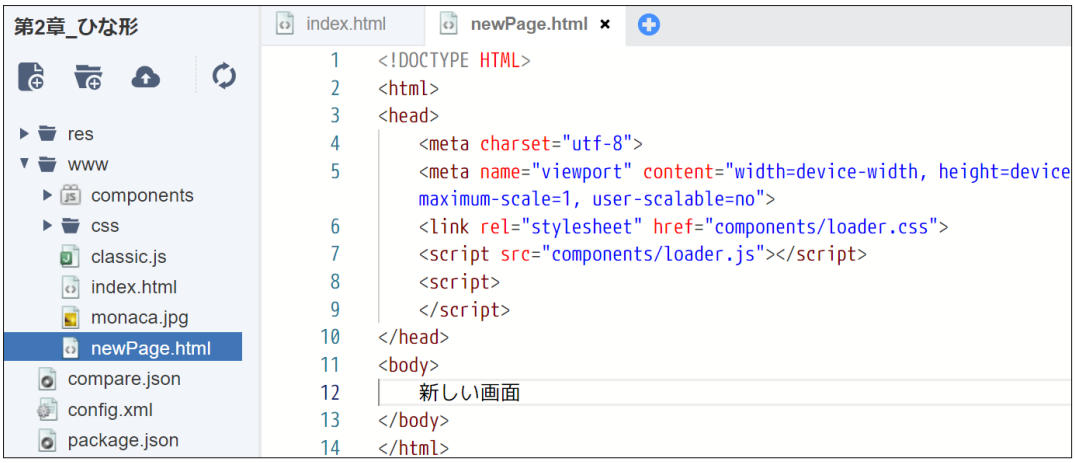
・同一フォルダ内のページに移動する場合はファイル名の指定だけで良い。
・フォルダとフォルダの区切り、またはフォルダとファイルの区切り文字として、フォルダ名の後ろにスラッシュをつける。
・一つ上のフォルダは .. という記号で表す。

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

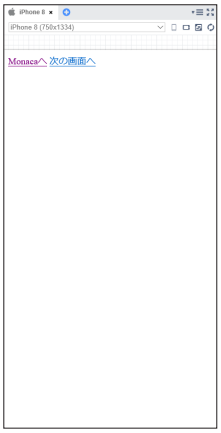
index.html

```
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>
```

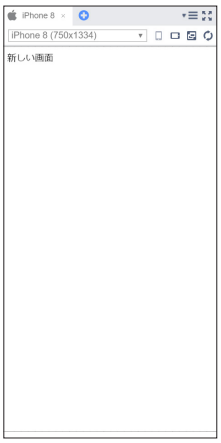
```
12  </head>
13  <body>
14      <a href="https://edu.monaca.io/">Monaca へ </a>
15      <a href="newPage.html"> 次の画面へ </a>
16  </body>
17  </html>
```

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

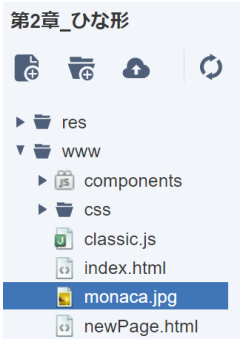
画面上に写真やイラストなどの画像を表示するには、HTML ファイルから画像ファイルを参照するようにタグで指定します。画像ファイルのパス（画像が置いてある場所）はリンクと同様、絶対パスまたは相対パスで指定します。

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。

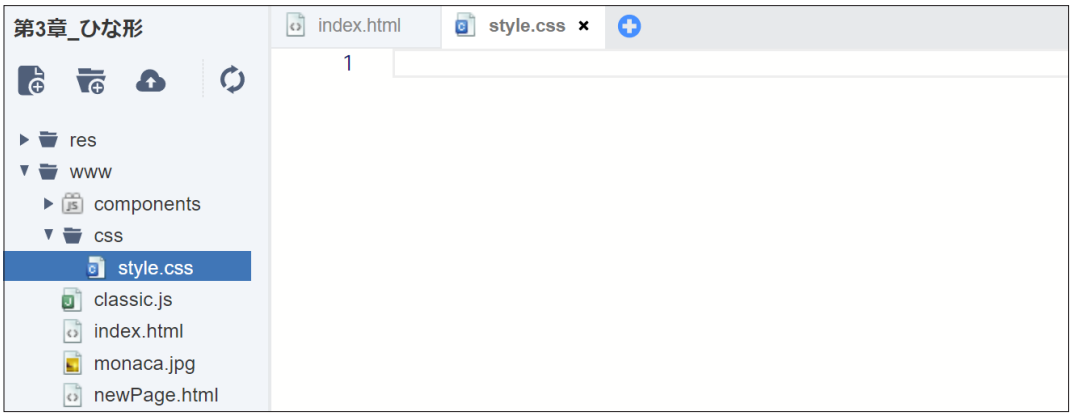
・タグセレクタ
　対象要素をタグ名で指定します。
・ID セレクタ
　対象要素を ID 属性値で指定します。ID 属性値は HTML 文書の中で一意となる（重複する値を設定できない）ので、特定の要素 1 つだけにスタイルを適用したい場合に利用します。
・クラスセレクタ
　CSS におけるクラスとは、スタイルをひとまとめにして名前をつけたもののことです。クラスはどのタグに対しても付けることができるので、複数の要素の中から任意の要素を選択してスタイルを適用したい場合に利用します。

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
```

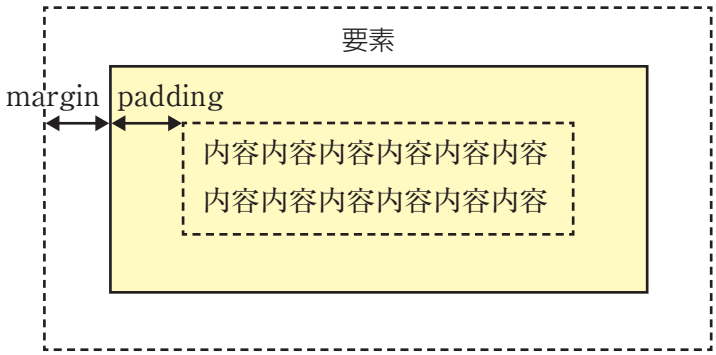
赤がff（255）、緑が00（0）、青がff（255）なので、原色の赤と青を混ぜた色＝紫になります。

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。

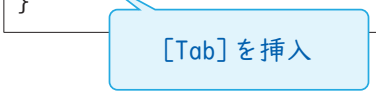
インデントとは

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

```
var 変数名;
```

例　xという名前の変数を作る

```
var 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

```
var 変数名 = 値;
```

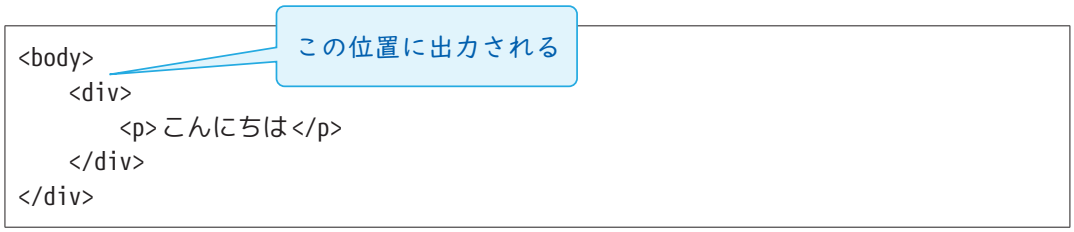
JavaScriptから画面にデータを出力する

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

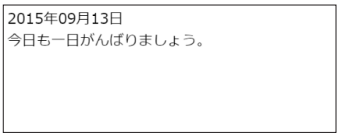
```
10      <script>
11          var today = "2015年09月13日";
12          document.write(today);
13      </script>
```

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>
        var 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

```
var 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          var date = new Date();
13          // 年、月、日の取得
14          var year = date.getFullYear();
15          var month = date.getMonth() + 1;
16          var day = date.getDate();
17          // 日本の表記にする
18          var 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。