Onsen UI

提供元：アシアル株式会社

目次

Onsen UI について

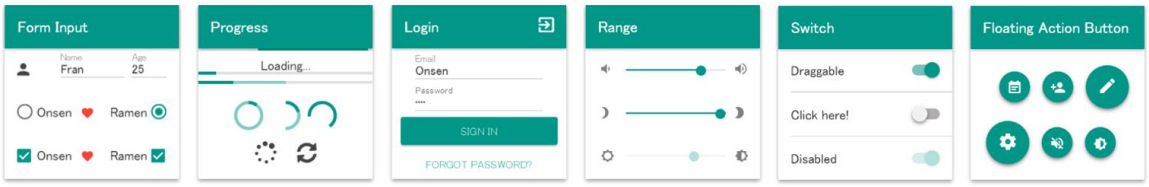
UI フレームワークとは

モバイルアプリを作成する場合、パソコン向けのアプリケーションとは異なり、アイコンを多用したり、ボタンを指で押しやすいサイズにしたりして、小さな画面の中にアプリの機能を盛り込む工夫が必要です。これらをすべて CSS や JavaScript で作り込むのは大変なので、一般的にアプリ開発の際には画面を作るためのソフトウェアである「UI フレームワーク」を利用します。

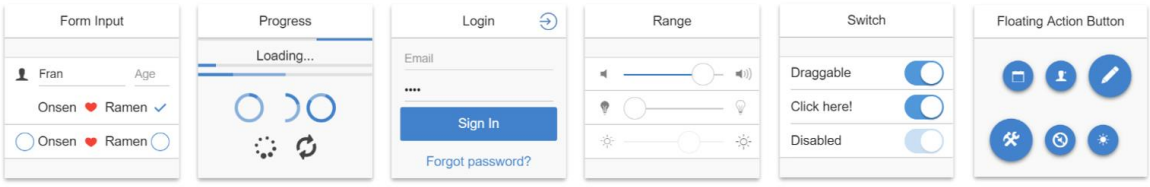
モバイルアプリのための UI フレームワーク、Onsen UI

Onsen UI は、独自の HTML タグを記述するだけで簡単にモバイルアプリの画面を作ることができる UI フレームワークです。ボタンやテキストボックスなどの入力項目や、ツールバーやタブバーといった画面のレイアウトを構成するパーツが豊富に用意されています。また、スマートフォン OS の種類を判別して、Android であればマテリアルデザイン（立体的な質感のデザイン）、iOS の場合はフラットデザイン（平坦な質感のデザイン）を自動的に適用する機能が搭載されています。

マテリアルデザイン（Android）



フラットデザイン（iOS）



詳細は https://ja.onsen.io/v2/ を確認してください。

サンプルアプリの紹介

Yes No チャート

いくつかの設問に「Yes」か「No」かを選択して回答していくと、最終的な診断結果が表示される性格診断アプリです。
このアプリは複数の HTML ファイルから構成されていて、１問回答するごとにアニメーションつきで次の画面へと遷移していきます。

プロジェクトの作成

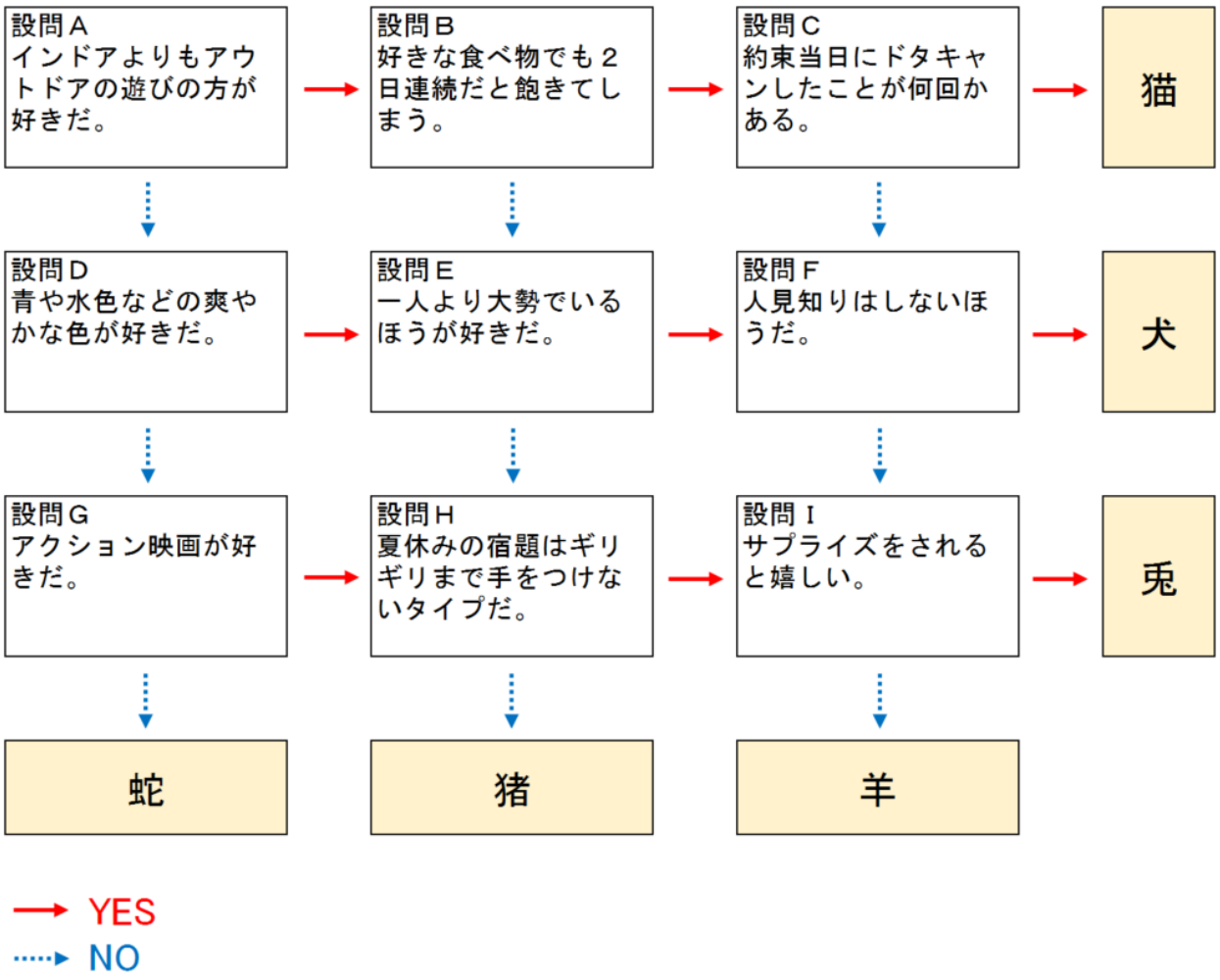
プロジェクトのインポート

インポート URL：https://ja.monaca.io/book/001/onsen-01a.zip

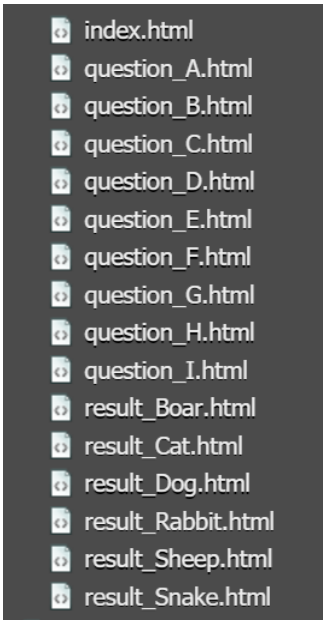
プロジェクト名：Yes No チャート

性格診断の流れ

設問は全 9 問、結果は 6 タイプです。以下のように画面が遷移します。



インポートしたプロジェクトでは、www フォルダの中に HTML ファイルが 16 枚入っています。



それぞれの役割は以下の通りです。

・index.html.....アプリの枠組み
・question_*.html .....設問 A～E
・result_*.html.....結果ページ

各ファイルの解説

1. index.html

```
16 <body>
17   <ons-navigator page="question_A.html" id="navi">
18   </ons-navigator>
19 </body>
```

Onsen UI では、様々なパーツを ons-で始まるタグで記述します。
index.html の<body>タグの中には、<ons-navigator>というタグのみが記述されています。このタグは複数の HTML ファイルを操作する役割を持つ特殊なタグで、アプリの画面上には表示されません。次のページに進んだり、前のページに戻ったりする機能を持っています。

【Onsen UI】複数ページの管理

```
<ons-navigator page="初期表示するページ"></ons-navigator>
```

17 行目で<ons-navigator>の page 属性に"question_A.html"を指定しているので、アプリを起動したときに最初に表示されるページは question_A.html になります。
また、id 属性に"navi"を指定しているので、"navi"という値で<ons-navigator>タグを JavaScript で操作できるようになります。

2. question_A.html（設問 A ページ）

```
1  <ons-page>
2    <ons-toolbar>
3      <div class="center">設問 A </div>
4    </ons-toolbar>
5
6    <div class="pagebody">
7      <h1 class="question">インドアよりもアウトドアの遊びの方が好きだ。</h1>
8      <ons-button class="yes-btn"
9        onclick="document.getElementById('navi').pushPage('question_B.html')">
10       YES
11     </ons-button>
12     <ons-button class="no-btn"
13       onclick="document.getElementById('navi').pushPage('question_D.html')">
14       NO
15     </ons-button>
16   </div>
17 </ons-page>
```

question_A.html では、3 つの Onsen UI タグが使われています。一番外側を囲んでいる<ons-page>タグは、Onsen UI でページを定義するために必要なタグです。アプリの画面に表示する内容を、このタグの中に記述します。

【Onsen UI】各ページの定義

```
<ons-page>ページ内に表示する内容</ons-page>
```

2～4 行目に記述されている<ons-toolbar>タグは、画面上部に表示されるツールバーです。<ons-toolbar>内には タグを配置します。 タグの class 属性に"left", "center", "right"のいずれかを指定して、ツールバー上に配置する横位置を決定します。
3 行目では タグの class 属性に"center"を指定しているので、ツールバーの中央に「設問 A」という文字列が表示されます。

【Onsen UI】ツールバー

```
<ons-toolbar>ツールバーに表示する内容</ons-toolbar>
```

8～11 行目と、12～15 行目に記述されている<ons-button>タグは、ボタンを表示するタグです。

【Onsen UI】ボタン

```
<ons-button>ボタンに表示する文字列</ons-button>
```

ここでは<ons-button>タグを使って YES ボタンと NO ボタンを表示しています。9 行目と 13 行目で、それぞれの onclick 属性にボタンを押したときの JavaScript の処理を設定しています。

ボタンが押されると、index.html に記述されている<ons-navigator>ボタンを使って、次のページに進む処理を行っています。
9 行目の YES ボタンが押されたときの処理は、以下のように記述されています。

```
document.getElementById('navi').pushPage('question_B.html')
```
index.html の<ons-navigator>タグ　　question_B.html に進む命令

pushPage という命令は、現在表示しているページから、次のページに進むための命令です。引数に指定した HTML ファイルを画面上に表示します。
この例では引数に question_B.html を指定しています。つまり、「設問 A で YES と答えた人には、設問 B を表示する」という処理を行っていることになります。設問 B のページでも同じように YES または NO の選択によって次のページに進み、最終的に result_で始まる結果ページの HTML ファイルが表示される、という流れになります。

3. question_B.html（設問 B ページ）

```
1  <ons-page>
2      <ons-toolbar>
3          <div class="left"><ons-back-button>Back</ons-back-button></div>
4          <div class="center">設問Ｂ</div>
5      </ons-toolbar>
6
7      <div class="pagebody">
8          <h1 class="question">好きな食べ物でも２日連続だと飽きてしまう。</h1>
9          <ons-button class="yes-btn"
10             onclick="">
11             YES
12         </ons-button>
13         <ons-button class="no-btn"
14             onclick="">
15             NO
16         </ons-button>
17     </div>
18 </ons-page>
```

question_B.html から question_I.html までのファイルは、question_A.html と一箇所だけ異なる箇所があります。3 行目の<ons-back-button>タグです。このタグは、前のページに戻るボタンを表示します。このボタンによって、一つ前のページに戻って YES または NO の選択をやり直すことができるようになります。

【Onsen UI】前のページに戻るボタン

```
<ons-back-button>ボタンに表示する文字列</ons-back-button>
```

4. result_Cat.html（結果ページ）

```
1  <ons-page>
2    <ons-toolbar>
3      <div class="center">診断結果</div>
4    </ons-toolbar>
5    <div class="pagebody">
6      <h1>猫タイプ</h1>
7      <img src="images/cat.png">
8      <p>猫タイプのあなたは、何ものにも縛られない自由人。気まぐれで自由奔放な性格に、周囲の人は振り回されているかも？</p>
9      <ons-button class="retry-btn" onclick="document.getElementById('navi').resetToPage('question_A.html')">
10       もう一度
11     </ons-button>
12   </div>
13 </ons-page>
```

結果ページの HTML ファイルは、設問ページの HTML とほぼ同じ構成になっています。一点だけ異なるのは、「もう一度」ボタンが押されたときの処理内容です。9 行目にボタンが押されたときに実行する JavaScript が記述されています。

```
document.getElementById('navi').resetToPage('question_A.html')
```

index.html の<ons-navigator>タグ　　ページ履歴をリセットし、TOP に戻る

resetToPage という命令は、TOP ページに戻るときに使います。pushPage とは異なってこれまでに表示したページの履歴がリセットされるので、戻るボタンで前のページに戻ることができなくなります。

このようにして、複数のページから構成されるアプリを作ることができます。

実習

question_B.html、question_C.html、（…中略…）、question_I.html までの 8 つの HTML ファイルは、ソースコードが虫食い状態になっています。YES または NO ボタンが押されたときの処理を追加して、アプリを完成させてください。設問の進め方は 3 ページ目を参考にしてください。

道路標識暗記アプリ

道路標識の種類を覚えるための学習補助アプリです。
リスト形式で一覧表示された道路標識の名前をタップすると、詳細ページに切り替わって道路標識の画像が表示されます。

プロジェクトの作成

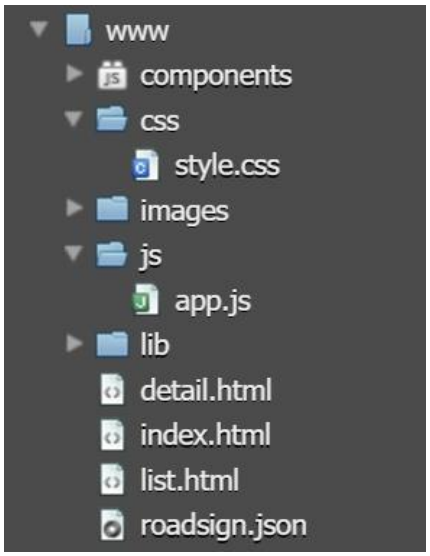
プロジェクトのインポート

インポート URL : https://ja.monaca.io/book/001/onsen-02a.zip

プロジェクト名：道路標識暗記アプリ

今回のアプリでは、道路標識データを JSON という形式のファイルに保存しておき、JavaScript で読み出します。そのため JSON を取得するための命令を持つ「jQuery」というソフトウェアが必要になります。インポートしたプロジェクトでは、あらかじめ jQuery が有効化されて使える状態になっています。

フォルダ構成は以下のようになっています。



それぞれの役割は以下の通りです。

・css/style.css ..... CSS ファイル

・images フォルダ ..... 道路標識の画像群

・js/app.js..... JavaScript ファイル

・lib フォルダ..... Onsen UI のライブラリ等

・detail.html..... 道路標識詳細ページ

・index.html ..... アプリの枠組み

・list.html ..... 道路標識一覧ページ

・roadsign.json ..... 道路標識データ

各ファイルの解説

1. index.html

```
<body>
    <ons-navigator page="list.html" id="navi"></ons-navigator>
</body>
```

今回のアプリは一覧ページと詳細ページの 2 つのページから構成されますので、<ons-navigator>タグを使って複数ページを管理しています。ここでは、最初に表示されるページは list.html であることを示しています。

2. list.html（一覧ページ）

```
<ons-page id="list-page">
    <ons-toolbar>
        <div class="center">道路標識一覧</div>
    </ons-toolbar>

    <ons-list id="signlist">
    </ons-list>
</ons-page>
```

list.html では、ツールバーの他には<ons-list>というタグのみが配置されています。<ons-list>は、リスト形式でデータを一覧表示するためのタグです。ここには記載されていませんが、リスト内の一件一件の項目は<ons-list-item>タグで表します。

【Onsen UI】リストの表示

```
<ons-list>
    <ons-list-item>項目１</ons-list-item>
    <ons-list-item>項目２</ons-list-item>
    <ons-list-item>項目３</ons-list-item>
</ons-list>
```

3. detail.html（詳細ページ）

```
<ons-page id="detail-page">
    <ons-toolbar>
        <div class="left"><ons-back-button>戻る</ons-back-button></div>
        <div class="center">道路標識詳細</div>
    </ons-toolbar>

    <h1 id="sign_name"></h1>
    <img src="" id="sign_image">
</ons-page>
```

詳細ページには、道路標識の名前を表示するための<h1>タグと、道路標識の画像を表示するための<img>タグが配置されています。一覧ページで選択された道路標識のデータを受け取って表示します。

4. roadsign.json（道路標識データ）

JSON（JavaScript Object Notation）は、データの形式の一種です。JavaScript から操作しやすいデータ形式ですので、アプリで大量のデータを扱う場合は、JSON 形式でデータを作成しておくのが良いでしょう。

JSON の形式は JavaScript の配列またはオブジェクト（連想配列）とほぼ同じですが、必ずプロパティ名をダブルクォートで囲む必要があります。シングルクォートは使えないので注意してください。

roadsign.json は、以下のように記述されています。

```
[
        {
                "name": "通行止め",
                "image": "1.png"
        },
        {
                "name": "車両通行止め",
                "image": "2.png"
        },
        …以降省略…
]
```

1 件分の道路標識データ

道路標識の名前（name）と画像（image）を 1 セットとして、道路標識データを表現しています。[]（角かっこ）は配列を表していますから、配列の中に複数の道路標識データが含まれている、という構造になっています。

5. js/app.js（JavaScript ファイル）

```
1  // 初期処理
2  ons.ready(showListData);
3
4  // 一覧ページを表示
5  function showListData() {
6
7
8  }
9
10 // 詳細ページを表示
11 function onItemClick(item) {
12
13
14 }
```

ファイルを開くと、処理が虫食い状態になっています。
2 行目に、ons.ready という命令が記述されています。この命令は、Onsen UI のタグがブラウザに解釈されて、JavaScript から操作可能になったタイミングで実行されます。アプリを起動して最初に実行したい関数をこの命令の中に記述します。今回は、一覧ページを表示するための関数を指定しています。

【Onsen UI】初期処理

```
ons.ready(最初に実行する関数)
```

実習

まずは、一覧ページを表示する showListData 関数を完成させます。

```
// 一覧ページを表示
function showListData() {
    // 道路標識データを取得
    $.getJSON("roadsign.json", function(result) {
        var count = result.length;
        var html = "";
        for(var i=0; i<count; i++) {
            // 道路標識の件数分、<ons-list-item>を連結したＨＴＭＬ文字列を作成
            html += "<ons-list-item onclick='onItemClick(" + JSON.stringify(result[i]) + ")'>"
            + result[i].name
            + "</ons-list-item>";
        }
        // 作成したＨＴＭＬ文字列を<ons-list>タグ内に挿入
        document.getElementById("signlist").innerHTML = html;
    });
}
```

roadsign.json に記述されている道路標識データを得るために、jQuery の命令である$.getJSON を使っています。

JSON データの取得

```
$.getJSON(ファイルの URL, データ取得後に実行する関数)
```

第一引数に指定した JSON ファイルの内容を取得し、取得が完了したら第二引数の関数を実行します。今回の例では、JSON ファイル取得完了後に以下の網掛けの箇所が実行され、引数 result の中に roadsign.json ファイルの中身が入ってきます。

```
// 一覧ページを表示
function showListData() {
    // 道路標識データを取得
    $.getJSON("roadsign.json", function(result) {
        var count = result.length;
        var html = "";
```

roadsign.json の中身

```
        for(var i=0; i<count; i++) {
            // 道路標識の件数分、<ons-list-item>を連結したＨＴＭＬ文字列を作成
            html += "<ons-list-item onclick='onItemClick(" + JSON.stringify(result[i]) + ")'>"
            + result[i].name
            + "</ons-list-item>";
        }
        // 作成したＨＴＭＬ文字列を<ons-list>タグ内に挿入
        document.getElementById("signlist").innerHTML = html;
    });
}
```

roadsign.json の中身は、複数の道路標識データが集まった配列になっています。関数の中で、配列の要素の数だけ for 文を繰り返し、道路標識の件数分<ons-list-item>タグが連なった HTML 文字列を作成しています。

```
// 道路標識の件数分、<ons-list-item>を連結したＨＴＭＬ文字列を作成
html += "<ons-list-item onclick='onItemClick(" + JSON.stringify(result[i]) + ")'>"
+ result[i].name
+ "</ons-list-item>";
```

+演算子が何度も出てきて少し複雑に見えますが、ここでは文字列と変数の中身を連結させているだけです。

result[i]には現在処理をしている i 番目の道路標識データが入っています。result[i].name とすると、道路標識の名前を参照することができます。

JSON.stringify という命令は、JavaScript の配列やオブジェクト（連想配列）を、文字列に変換する命令です。文字列データを作成する場合はこの命令を使います。

配列やオブジェクトを文字列に変換する

```
JSON.stringify(配列やオブジェクト)
```

1 件分の繰り返しが終わると、以下のような文字列が作成されます。

```
<ons-list-item onclick='onItemClick({"name":"通行止め","image":"1.png"})'>通行止め</ons-list-item>
```

すべての繰り返しが終わったら、最後に list.html の<ons-list>タグ内に HTML 文字列を挿入しています。最終的には、list.html 内の HTML は以下のようになります。

```
<ons-list id="signlist">
    <ons-list-item onclick='onItemClick({"name":"通行止め","image":"1.png"})'>通行止め</ons-list-item>
    <ons-list-item onclick='onItemClick({"name":"車両通行止め","image":"2.png"})'>車両通行止め</ons-list-item>
    <ons-list-item onclick='onItemClick({"name":"車両進入禁止","image":"3.png"})'>車両進入禁止</ons-list-item>
    …以下省略…
</ons-list>
```

ここまでできたら、アプリ起動時に道路標識一覧が表示されるようになります。

つづいて、リストの項目をタップしたときに詳細ページに移る処理を追加します。onItemClick 関数の中に以下のように記述して下さい。

```
// 詳細ページを表示
function onItemClick(item) {
    document.getElementById("navi").pushPage("detail.html")
    .then(function() {
        // 道路標識の名称と画像ファイルを画面に埋め込む
        document.getElementById("sign_name").innerHTML = item.name;
        document.getElementById("sign_image").src = 'images/' + item.image;
    });
}
```

この onItemClick 関数は、一覧画面を作成したときに設定したクリックイベントから呼び出されます。引数として渡された道路標識データは、item 変数の中に代入されます。

```
<ons-list-item onclick='onItemClick({"name":"通行止め","image":"1.png"})'>通行止め</ons-list-item>
```

```
function onItemClick(item) {
    document.getElementById("navi").pushPage("detail.html")
    .then(function() {
        // 道路標識の名称と画像ファイルを画面に埋め込む
        document.getElementById("sign_name").innerHTML = item.name;
        document.getElementById("sign_image").src = 'images/' + item.image;
    });
}
```

呼び出された onItemClick 関数では、<ons-navigator>の pushPage 命令を使って、詳細ページである detail.html へ移動しています。pushPage に then という命令を繋げて記述することで、ページが移動した直後に任意の処理を実行させることができます。通常は、移動先ページの初期処理などを記述します。

次のページに進み、移動後に何らかの処理を行う

```
ons-navigator 要素.pushPage(移動先ページ).then(ページ移動後に実行する関数);
```

```
function onItemClick(item) {
    document.getElementById("navi").pushPage("detail.html")
    .then(function() {
        // 道路標識の名称と画像ファイルを画面に埋め込む
        document.getElementById("sign_name").innerHTML = item.name;
        document.getElementById("sign_image").src = 'images/' + item.image;
    });
}
```

詳細ページに移動した直後に実行する処理

ここでは、引数として受け取った道路標識の名前と画像を、詳細ページに埋め込んでいます。
以上で、道路標識暗記アプリは完成です。

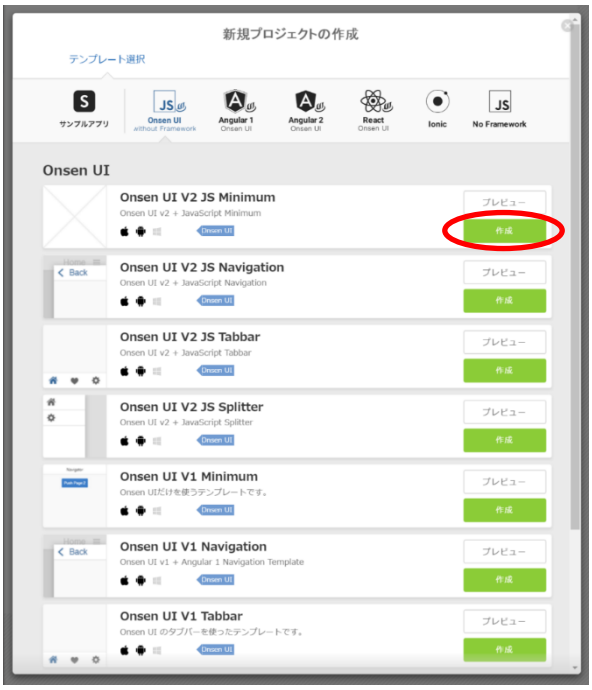
その他の使い方

新規プロジェクトの作成方法

ご紹介したサンプルアプリはプロジェクトをインポートして作成しましたが、新規で Onsen UI を利用したアプリを開発する場合は、以下のような手順でプロジェクトを作成します。

1. Monaca にログインし、「新規プロジェクトの作成」ボタンをクリックしてテンプレート一覧を表示します。

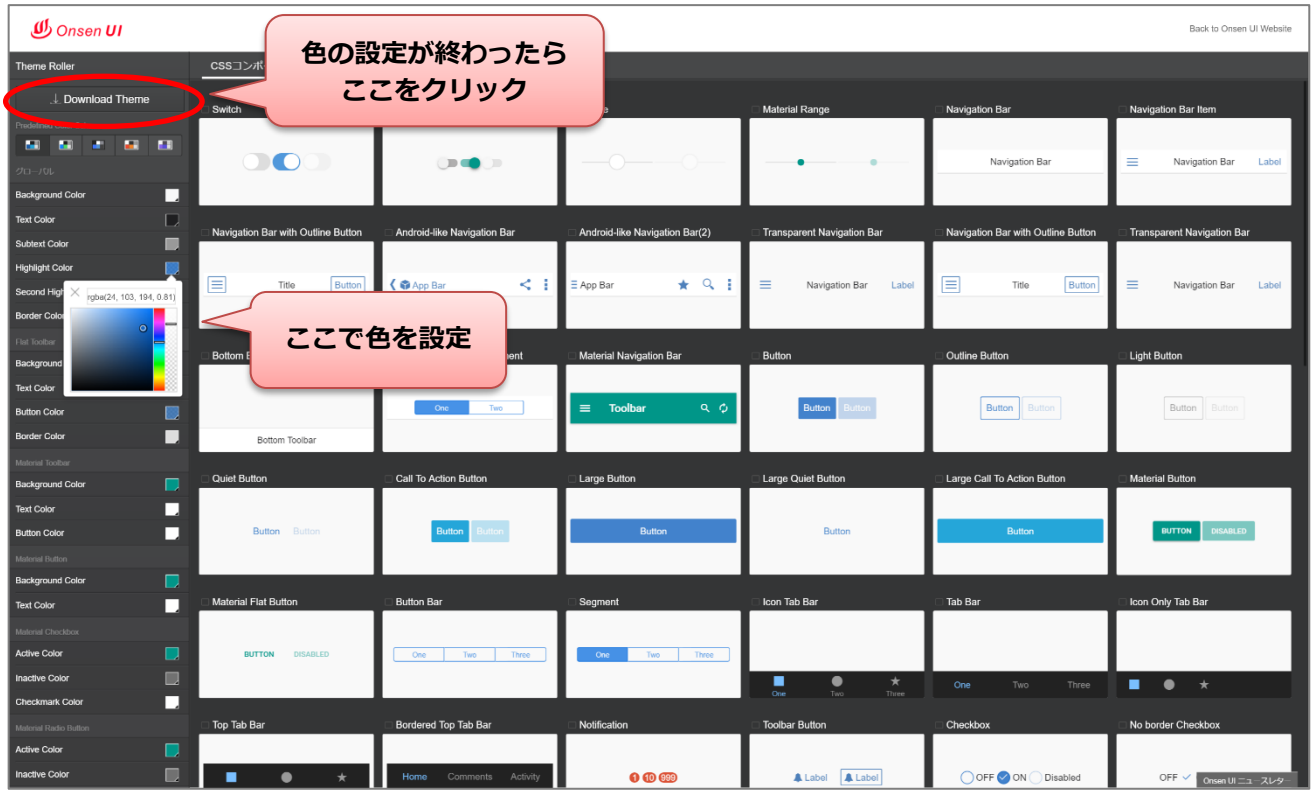
2. 「Onsen UI V2 JS Minimum」テンプレートの「作成」ボタンをクリックし、プロジェクト名を入力したら作成完了です。



テーマカラーを変える方法

Onsen UI 標準のテーマカラーは、Android は緑、iOS は青になっていますが、「テーマローラー」を使って色を自由に変えることができます。

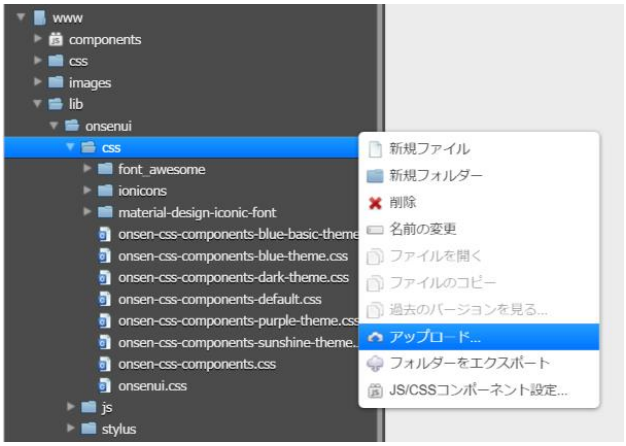
Onsen UI テーマローラー
http://components2.onsen.io/



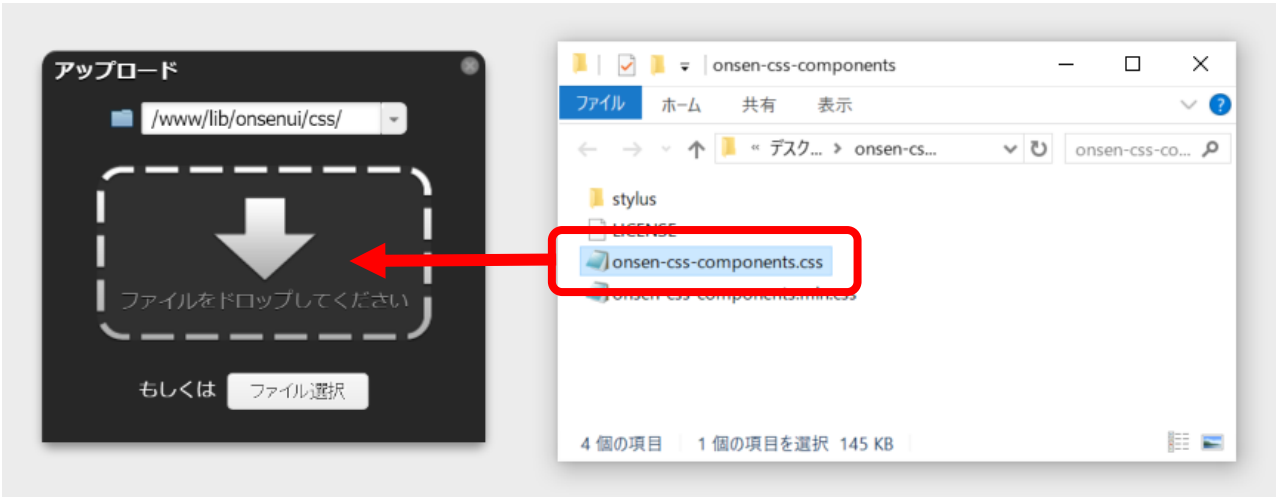
画面の左側で、各パーツの色を設定できます。設定が完了したら、「Download Theme」ボタンを押してください。「onsen-css-components.zip」という名前のファイルがダウンロードされます。
ファイルを解凍（Windows パソコンの場合、ファイルを右クリックして「すべて展開」を選択）すると、以下のファイルが表示されます。

この中の、「onsen-css-components.css」を Monaca にアップロードすれば、テーマカラーが変更されます。
アップロード手順は次の通りです。

1. www/lib/onsenui/css フォルダを右クリックし、「アップロード」を選択します。



2. アップロード画面の点線の枠内に、「onsen-css-components.css」をドラッグ＆ドロップします。



3. 上書き確認のメッセージが表示されるので、「OK」を押したら完了です。テーマカラーが変更されていることを確認して下さい。