SECTION 01

Vue.jsについて理解しよう

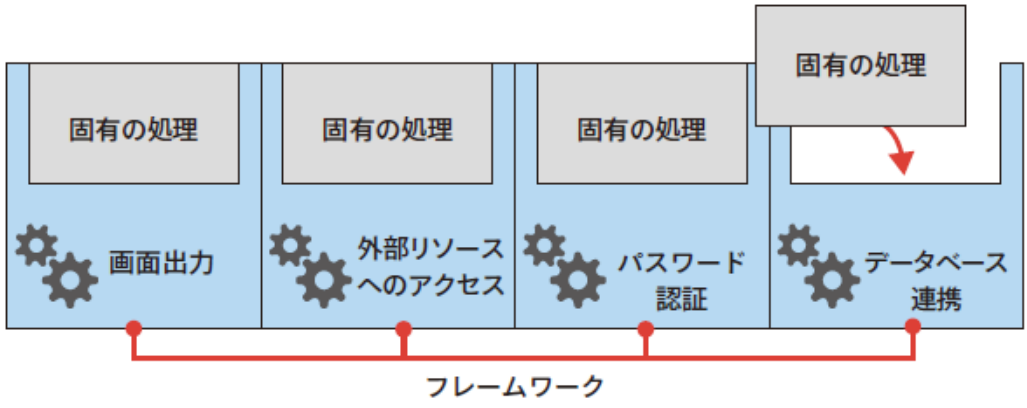
Vue.jsは、現在フロントエンドの領域で非常に高い人気を誇るJavaScriptフレームワークです。まずはVue.jsの人気の理由であるプログレッシブフレームワークやコンポーネント指向といったキーワードの意味を説明します。

◎ なぜフレームワークが必要？

今日では、さまざまなサービスのためのアプリケーションが開発されていますが、外部リソースへのアクセスや操作画面へのデータの反映などといった処理は、多くのアプリケーションに共通する処理です。フレームワークはこうした標準的な処理をあらかじめ「骨組み」として用意し、開発者はそれ以外に必要な固有の処理を書き足します。これにより、開発工数の大幅な削減が図れます。また、開発者はフレームワークのルールに従ってコードを書くことになるので、書き方が統一され、バグを減らしたり、発見を容易にしたりすることにも繋がります。

フレームワークを使わなくてもアプリケーションの開発は可能ですが、開発・保守を大幅に効率化できるため、現在のアプリケーション開発ではフレームワークを使って開発することがほとんどです。

図1-1 フレームワークのイメージ（一例）



◎ エディタとVue DevToolsのセットアップ

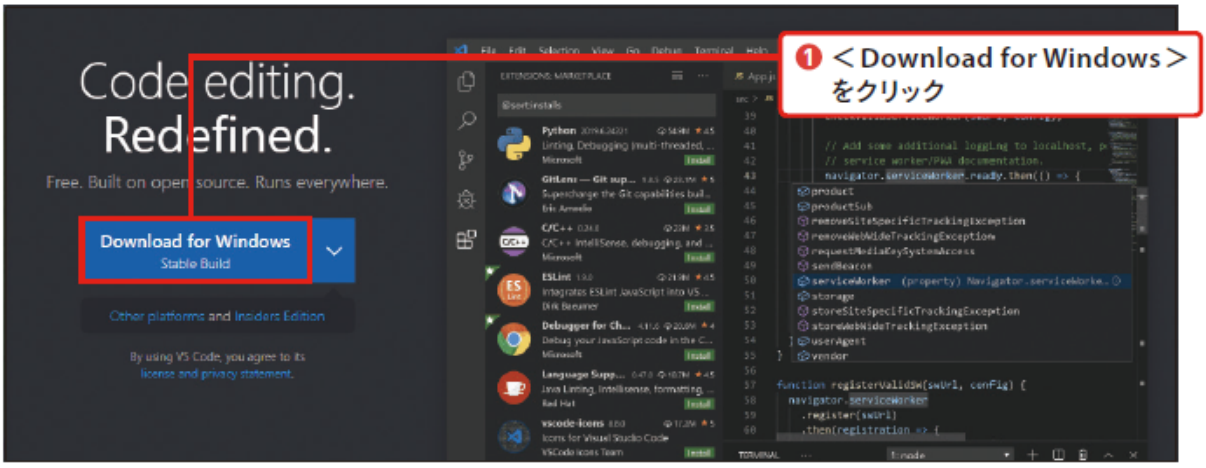
Visual Studio Codeのダウンロード

エディタは好みのものでよいですが、開発に便利なプラグインが多数用意されている、Microsoftの Visual Studio Codeがおすすめです。早速ダウンロードしてみましょう。ブラウザで下記のダウンロードページにアクセスしてください。

- Visual Studio Codeのダウンロードページ
  https://code.visualstudio.com/

TOPにある＜Download for Windows＞（macOSの場合は＜Download for Mac＞）をクリックすると❶、ダウンロードが始まります。

図1-7 Visual Studio Codeのダウンロードページ



Visual Studio Codeのインストール

ダウンロードが完了したら（デフォルト状態ではユーザーのDownloadsフォルダに保存されます）、ダウンロードしたインストーラーを実行します。セットアップウィザードが開始するので、使用許諾契約書、インストール先を確認し、先に進みます❷❸❹。

リスト1-1 index.html

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width,initial-scale=1.0" />
    <title>sample</title>
  </head>
  <body>
  </body>
</html>
```

Vue.jsを有効にする要素にidを指定する

次に、HTMLの要素に対しVue.jsの機能を適用させる準備をします。body要素の直下にdiv要素を作成し、divタグにidを振ります。idの値は任意の文字列でかまいませんが、今回はid="app"とします❼。JavaScript側では、このidをターゲットとして、Vue.jsが提供するデータやメソッドを使えるようにするので、必ず記述します。また、**タグに記述したidと、JavaScript側で指定するidが1文字でも違うとVue.jsは適用されない**ので、間違えないように入力しましょう。

リスト1-2 idの設定（index.html）

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width,initial-scale=1.0" />
    <title>sample</title>
  </head>
  <body>
    <div id="app">hello Vue.js</div>
  </body>
</html>
```

❼ idの設定

Vue.js本体とアプリケーションのJavaScriptファイルを読み込む

次に、Vue.js本体と、アプリケーションのコードを記述するJavaScriptファイルを、HTMLで読み込むための準備をします。

body要素の後に、下記のようにscript要素を2つ追記します。1つ目はVue.js本体のCDNを読み込みます❽。2つ目はこれから作成するVue.jsアプリケーションのJavaScriptファイルを読み込みます❾。

POINT

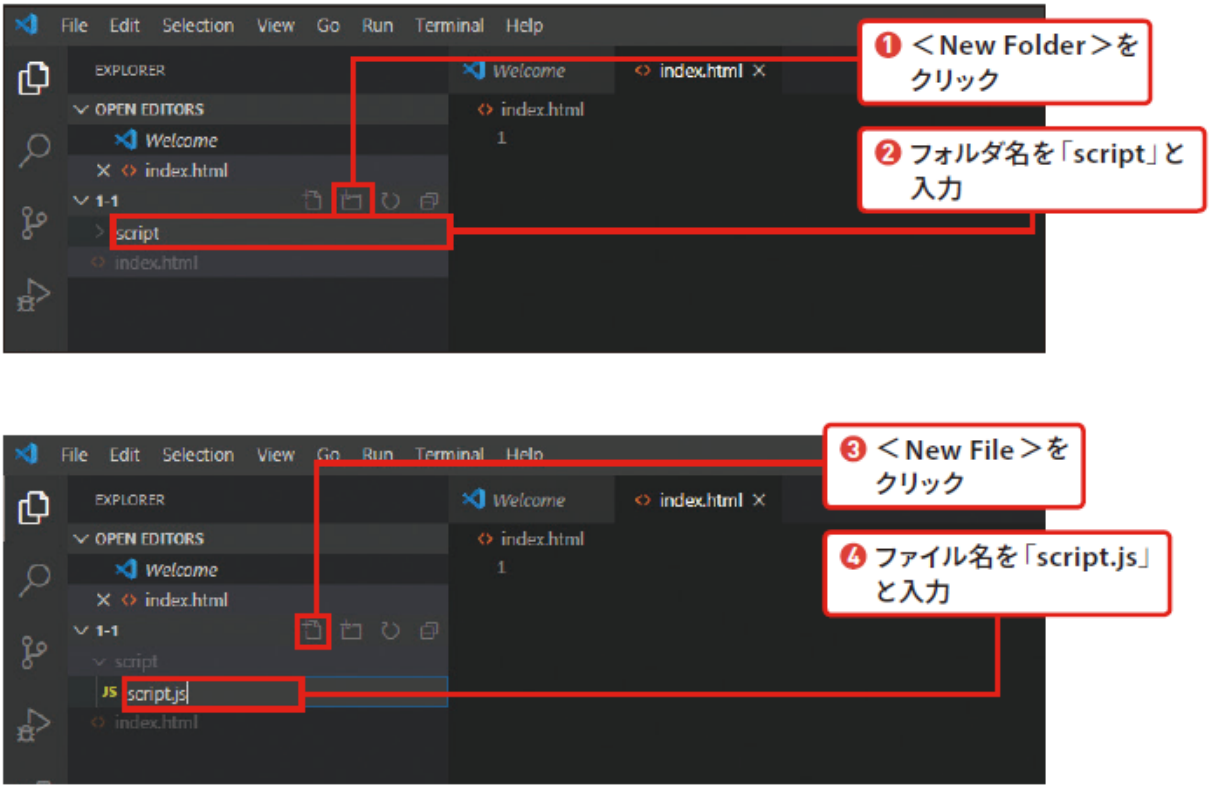
本書ではCDNを読み込む方法で進めていきますが、https://jp.vuejs.org/v2/guide/installation.htmlからJavaScriptファイルをダウンロードし、ローカルに設置して利用することもできます。

◎ JavaScriptファイルの準備

次に、JavaScriptファイルの準備です。これがVue.jsアプリケーションの核ともいえる部分です。先ほどHTML側のscript要素で指定したとおり、1-1/scriptフォルダの中にscript.jsを作成します。

＜New Folder＞をクリックし❶、1-1/scriptフォルダを作成します❷。さらに＜New File＞をクリックして❸、その中にscript.jsを作成します❹。

図1-22 1-1/scriptフォルダの作成



作成したscript.jsに、リスト1-4のように記述してください。

SECTION

03 バインディングを理解しよう

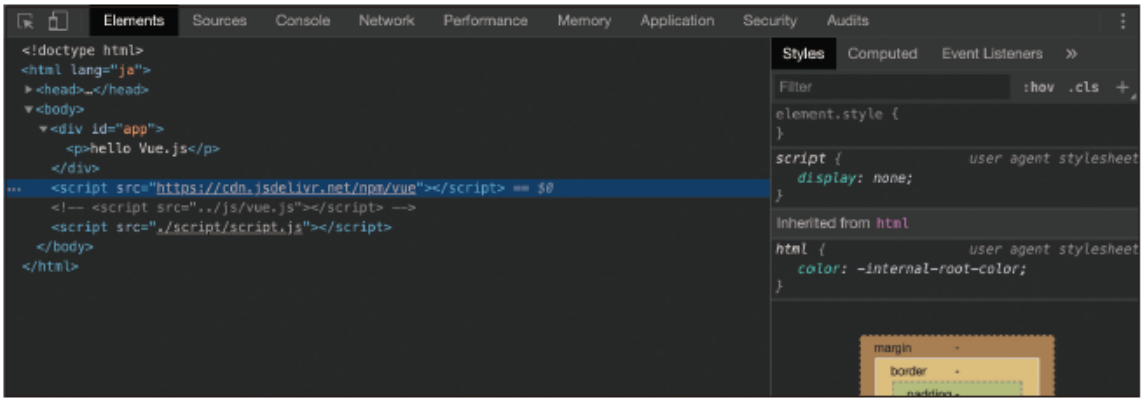
前SECTIONのアプリケーションは、Vueインスタンスのデータを表示するだけで、Vue.jsを使う意味があまり感じられなかったかもしれません。しかし、実はVue.jsを理解するに当たり重要なバインディングという仕組みが使われていました。ここでは、通常のJavaSctiptでHTMLの内容を書き換える場合とも比較しながら、バインディングのメリットを理解しましょう。

◎ DevToolsでVueインスタンスのデータを書き換えてみよう

バインディングの説明に入る前に、先ほど作ったサンプルの、textプロパティのデータをDevToolsで書き換えてみましょう。

先ほどのサンプルをGoogle Chromeで表示した状態のまま、Windowsの場合はF12、macの場合はcommand+option+iを押し、DevToolsを開きます。

図2-5 DevToolsの画面



DevToolsのConsoleタブのエリアでは、任意のJavaScriptコードを実行することができます。この機能を使ってVueインスタンスのデータを書き換えてみます。DevToolsの画面上部の＜Console＞タブをクリックしてConsoleパネルに切り替え❶、そこにapp.text = 'goodbye!!'と入力し、Enterを押

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

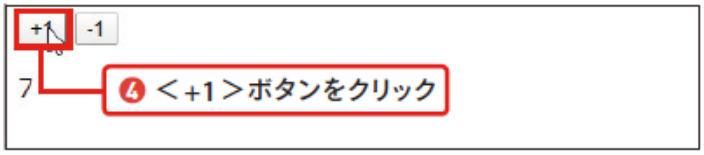
サンプルを実行してみよう

それでは、ブラウザでindex.htmlを表示してみましょう。以下のように、2つのボタンと、初期値の0が表示されます。

図3-4 初期表示

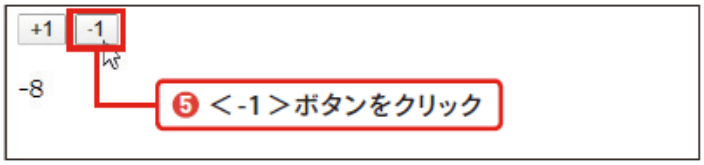
＜+1＞ボタンをクリックしてみてましょう❹。クリックするたびに値が1ずつ増えていけば成功です。

図3-5 値を増やす



次に＜-1＞ボタンをクリックしてみてください❺。クリックするたびに値が1ずつ減っていけば成功です。

図3-6 値を減らす



Vue DevToolsでの確認

ここで、Vue DevToolsを使って値を確認してみましょう。まず、DevToolsを開きます（Windowsの場合はF12、Macはcommand＋option＋i）❶❷。

SECTION 01

条件分岐と条件付きレンダリングを理解しよう

Web アプリケーションでは、条件によって画面に表示させる要素を変えたり、表示・非表示を切り替えたりしたい場合があります。そうした場合に使うのが条件付きレンダリングです。このSECTION ではまず、プログラムの基本構造の 1 つである条件分岐というものについて説明し、さらに Vue.js での条件付きレンダリングの使い方を解説します。

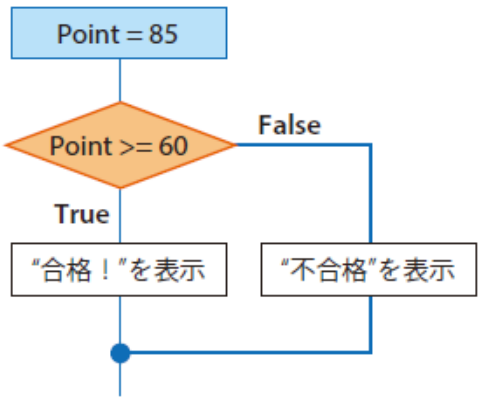
◎ 条件分岐

◎ 条件分岐とは

条件分岐とは、プログラムの基本的な制御構造の1つで、条件によって、実行させる処理を変えることです。条件に当てはまるかどうかを判定する、条件式の真偽値が**真（true）**なのか、**偽（false）**なのかによって、片方の処理を実行し、もう片方の処理を無視します。JavaScript では、条件分岐を行うために、if 文や switch 文などが使われます。

以下の図を見てください。左は点数が 60 点以上か否かで「合格！」か「不合格」のいずれかを表示する条件分岐を示した図です。このような流れを表す図を**フローチャート**と呼びます。

図 4-1 条件分岐のフローチャート



SECTION 03

ナビゲーションメニューを表示しよう

次は、リストレンダリングの実践的な使い方として、Webページのナビゲーションメニューを作ってみましょう。先ほど作成したオブジェクトの配列からのリストレンダリングの応用で、リンク先などの情報をオブジェクトのプロパティにします。

◎ ナビゲーションメニューを作ってみよう

ここでは、リストレンダリングを使い、以下のようなナビゲーションメニューを作成してみましょう。メニューからは、各ページにリンクできるようにします。

図5-8 ナビゲーションメニュー

TOP　ABOUT　SCHEDULE　CONTACT

ナビゲーションメニューは他のHTMLファイルにリンクするので、以下のものが必要です。

- **表示用ラベル**
- **リンク用のURL文字列**
- **リンク先のHTMLファイル**

新たに作業用フォルダ（「5-2」フォルダ）を用意しておきましょう。

リスト6-4 オブジェクト指定でのスタイルバインディング

```
<div :style="{ width: width + 'px', height: height + 'px', backgroundColor: bgColor}"></
div>
...
<script type="text/javascript">
  ...
  data() {
    return {
      width: 100,
      height: 200,
      bgColor: "#fcfcfa"
    };
  }
  ...
</script>
```

POINT

CSSプロパティ名を一般的なケバブケース（すべて小文字で、単語の間をハイフンで区切る書き方）で記述する場合は、'background-color'のように、シングルクォートで囲む必要があります。

◎ 色を変化させるアプリを作ろう

スタイルバインディングの実践的な学習として、画像をリアルタイムで変化させるアプリを作ってみましょう。4つのスライダーを用意し、これを調節することで画面に表示される正方形の大きさや色を自由に変更できるようにします。

新たに作業用フォルダ（「6-1」フォルダ）を用意しておきます。

図6-1 サンプルの完成イメージ

未完了TODOが登録された状態です。

図7-11 未完了TODOを登録

いくつかTODOを追加したり＜DONE＞ボタンをクリックして完了TODOエリアに移動させたりしてみましょう。

図7-12 TODOを完了TODOエリアに移動

＜DELETE＞ボタンをクリックするとTODOが削除されることも確認しましょう。

図7-13 TODOの削除

◎ 動作を確認しよう

それでは、index.htmlをブラウザで開いてみましょう。まず、先に作成したサムネイル画面が表示されます。

図8-11 サムネイル画面の表示

次に、サムネイルをどれかクリックしてみましょう。モーダル画面が開き、クリックした画像が大きく表示されます。画像をクリックすると閉じます。

図8-12 モーダル画面の表示