ワンポイント thisの意味は利用する場面で変わる

P.173で使用した「this」は状況によって指し示すものが変わる特殊なキーワードです。大きく分けると、関数の中で使うthisと、関数の外で使うthisで表すものが違います。

関数定義の外で使ったthisは、windowオブジェクトを指します。以下のように、比較演算子で厳密な比較を行っても真（true）になります。

メソッドの定義内で使ったthisは、メソッドが所属するオブジェクトを示しています。Lesson 46では、omikujiオブジェクトのgetResult()メソッドから、同じomikujiオブジェクトのresultsプロパティを利用するためにthisを利用しました。このとき、thisはomikujiオブジェクトを指しています。

上級者向けの内容になるので本書では扱いませんが、関数内のthisは、関数の呼び出し方によってもthisの値が指すものが変化します。メソッド定義内で使用した場合以外にもいくつかパターンがあるのです。さらに詳しくthisについて知りたい場合は、Chapter 13で紹介しているMDNのリファレンスで調べてみるといいでしょう。

▶ 関数の外部で使うthisはwindowオブジェクトを指す

```
console.log(this === window);
```
…… 厳密に等しいので結果はtrue

▶ メソッド定義の中で使うthisは所属するオブジェクトを指す

```
// おみくじオブジェクトの定義
var omikuji = {
  results: ["大吉","吉","中吉","小吉","凶"],
  getResult: function() {
    var results = this.results;
    return results[Math.floor(Math.random() * results.length)];
  }
}
```

このthisはomikujiオブジェクトを指す

JavaScriptのthisの働きは非常に奥が深いのですが、とりあえず関数の内部と外部でthisの値が異なるということだけ理解しておいてください。

Chapter

9

フォトギャラリーを作成しよう

この章では、これまでの学習の集大成として、フォトギャラリーを作成します。実践を通じて学んだ内容を復習しながら、確かな力にしていきましょう。

Lesson 47 [ゴールの確認]

フォトギャラリーの設計を確認しましょう

このレッスンのポイント

この章では、これまでの学習の集大成として、「フォトギャラリー」を作成します。写真がメインコンテンツとなるWebサイトでよく見かけるパーツですね。まずは制作するフォトギャラリーの完成イメージを理解して、プログラミングの方針を立てていきましょう。

学んだことを活用しよう

皆さんはこれまでの章で、JavaScriptの基本的な知識を学びました。プログラミングのスキルを高めるには、こうした知識のインプットと、実際にプログラムを書いていくアウトプットが欠かせません。

この章では、これまでの学習の集大成として、以下のようなフォトギャラリーを作成しましょう。これまでに学んだ基本文法（Chapter 1〜2）、条件分岐（Chapter 3）、繰り返し処理（Chapter 4）、関数（Chapter 5）、HTML/CSSの操作（Chapter 6）、イベント（Chapter 7）、オブジェクト（Chapter 8）などの要素をフル活用します。

▶ フォトギャラリー

学んでインプットした知識を実際に使えるものにするには、たくさんのプログラムを書いて、アウトプットすることが大切です。

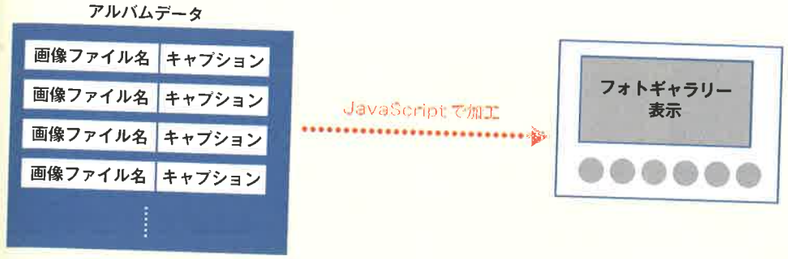
フォトギャラリーの仕様を確認する

今回作成するフォトギャラリーは、選択中の写真画像とそのキャプションを表示する「メイン」部分と、選択可能な写真画像が一覧表示される「サムネイル」部分から構成されます。

フォトギャラリーに読み込まれる写真画像とそのキャプションは、データにまとめてプログラムで管理できるようにします。このデータを「アルバムデータ」と呼ぶことにしましょう。

フォトギャラリーの使い勝手をよくするために、選択できるサムネイルの数は、「アルバムデータ」に登録されているデータの数によって自動的に変更されるようにしましょう。

▶ アルバムデータからHTMLを作る



▶ ファイル構成

プログラムを作成する前に、機能を図に整理しておくとプログラミングがスムーズになります。

Lesson 48 [HTML&の操作の実践]

アルバムデータからHTMLを作りましょう

このレッスンのポイント

まずは、アルバムデータを読み込んでフォトギャラリーのHTMLを作る処理を記述していきましょう。JavaScriptでHTMLを作るとHTMLの構造がイメージしづらいので、事前に最終的にできあがるHTMLのイメージも合わせて確認しておきましょう。

データからHTMLを作る理由

今回作成するフォトギャラリーは、アルバムデータのデータ数に応じて表示される写真画像の数も変わります。例えば、アルバムデータが1枚分しかなければ、サムネイルも1枚分しか表示されませんが、データが10枚分あれば、サムネイルも10枚分が表示されます。データによって内容が変わる部分をはじめからHTMLファイルに記述することはできないので、HTML側にはフォトギャラリーを表示するための「表示枠」だけを作り、データによって内容が変わる部分はJavaScriptを使って、データからHTMLを作るようにします。

▶ 最終的にできあがるHTMLのイメージ

```
__<div_id="gallery">
____<div_class="main">
______<img_src="img/1.jpg"_alt="山道の緑が気持ちいい">
______<p>山道の緑が気持ちいい</p>
____</div>
____<div_class="thumb">
______<img_src="img/1.jpg"_alt="山道の緑が気持ちいい">
______<img_src="img/2.jpg"_alt="階段きつかった">
______<img_src="img/3.jpg"_alt="高尾山薬王院！">
______<img_src="img/4.jpg"_alt="帰りはロープウェイでスイスイ">
______<img_src="img/5.jpg"_alt="〆のお蕎麦です">
____</div>
__</div>
```

JavaScriptで作成するメイン画像とキャプション

JavaScriptで作成するサムネイル

フォトギャラリーの表示枠を作る

1 HTMLファイルを編集する

09/gallery/practice/index.html

まずは、HTMLでフォトギャラリーの「メイン」と「サムネイル」を表示する枠を準備しましょう。このレッスンのindex.htmlファイルをBracketsで開いて、以下のコードを記述してください。

HTMLファイルに記述するのはアルバムデータの内容にかかわらず必要な部分だけです。まずは、フォトギャラリーの表示枠をdiv要素で作成します。その中に、メインとサムネイルを表示する枠をそれぞれdiv要素で作成します。

実際に表示する写真画像やキャプションは、アルバムデータを元にJavaScript側で作成します。

```
008 <body>
009 __<div_id="gallery">
010 ____<div_class="main">
011 ____</div>
012 ____<div_class="thumb">
013 ____</div>
014 __</div>
015 __<script_src="js/app.js"></script>
016 </body>
```

1 メインの枠を作成

2 サムネイルの枠を作成

ここで作成したのはフォトギャラリーを表示するための"枠"だけなので、この時点でブラウザに何も表示されていなくてOKです。

ワンポイント データのやりとりに便利な「カスタムデータ属性」

このサンプルでは写真画像のキャプションを記憶するためにalt属性を使用しますが、本来alt属性は「写真画像が表示されなかったときに表示する代替テキスト」を指定する属性です。

今回は写真画像に関連するキャプションを記憶したのでalt属性でも問題ありませんが、例えば「写真画像に関連する音楽ファイルの名前」といったデータを指定したい場合はどうしたらいいでしょうか。

HTML要素に対して、HTMLの仕様にない任意のデータを記憶させたい場合は、「カスタムデータ属性」というオリジナルの属性を作ります。

カスタムデータ属性では「data-xxx」という形式で好きな属性名を付けることができ、任意のデータをHTML要素に関連付けてセットすることができます。

```
setAttribute('data-bgm',_属性値); ····· カスタムデータ属性のセット
getAttribute('data-bgm'); ····· 属性値の参照
```

NEXT PAGE

アルバムデータを作る

1 JavaScriptファイルを編集する 09/gallery/practice/js/app.js

続いて、フォトギャラリーで読み込む「写真画像」と「キャプション」をまとめたアルバムデータを作ります。このレッスンのapp.jsファイルをBracketsで開いて、以下のコードを記述してください。

今回は、albumという配列の中に、写真画像の場所を示す「src」プロパティと、キャプションを示す「msg」プロパティを持ったオブジェクトを格納していきます❶。

```
001 //_アルバムデータの作成
002 var_album_=_[
003 __{src:_'img/1.jpg',_msg:_'山道の緑が気持ちいい'},
004 __{src:_'img/2.jpg',_msg:_'階段きつかった'},
005 __{src:_'img/3.jpg',_msg:_'高尾山薬王院！'},
006 __{src:_'img/4.jpg',_msg:_'帰りはロープウェイでスイスイ'},
007 __{src:_'img/5.jpg',_msg:_'〆のお蕎麦です'}
008 ];
```

❶ アルバムデータの配列を作成

Point オブジェクトを配列でまとめる

オブジェクトもデータの一種なので、配列でまとめることができます。また、配列もデータなので、オブジェクトの中に入れることもできます。

呼び出す際にプロパティ名が必要なものは「オブジェクト」、必要ないものは「配列」でまとめると、扱いやすいデータになります。

アルバムデータから最初の写真画像を表示する

1 メインの写真画像を準備する 09/gallery/practice/js/app.js

続いて、アルバムのデータからHTMLを作っていきます。

まず、フォトギャラリーを表示したときにアルバムの最初のデータを表示するようにしましょう。createElementメソッドで写真画像を表示するためのimg要素を作り、mainImageという変数に格納します❶。

次に、写真画像を表示するため、src属性に写真画像の場所を指定します。アルバムデータの最初のデータ「album[0]」から、写真画像の場所が記憶されているsrcプロパティ「album[0].src」を参照して、その値をセットします❷。

同様の手順で、alt属性にキャプションの値をセットします❸。

```
009
010 //_最初のデータを表示しておく
011 var_mainImage_=_document.createElement('img');
012 mainImage.setAttribute('src',_album[0].src);
013 mainImage.setAttribute('alt',_album[0].msg);
```

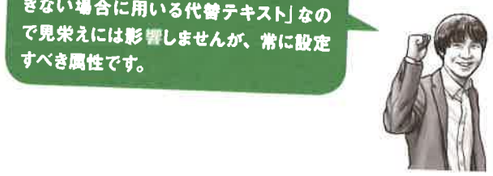
❶ img要素を作成
❷ src属性をセット
❸ alt属性をセット

2 メインのキャプションを準備する

同じように、メインのキャプションを用意します。p要素を作成し❶、メインの写真画像のalt属性のテキストを表示します❷。

```
014
015 var_mainMsg_=_document.createElement('p');
016 mainMsg.innerText_=_mainImage.alt;
```

❶ p要素を作成
❷ キャプションをセット



NEXT PAGE

3 HTMLに反映する

最後に、メイン画像とキャプションを表示する要素を取得してmainFlameという変数に格納し❶、insertBeforeメソッドを使ってHTMLに挿入します❷。

これでアルバムの最初のデータがメイン画像として表示されます❸。

```
017
018 var mainFlame = document.querySelector('#gallery .main');
019 mainFlame.insertBefore(mainImage, null);
020 mainFlame.insertBefore(mainMsg, null);
```

❶ 要素を取得
❷ 要素を追加

アルバムの最初のデータが表示された

今回は前処理が長かったので、一発で表示できた人はすごいです。表示できなかった場合は、コンソールにエラーが出ていないか確認してみましょう。

サムネイルを表示する

1 アルバムデータを読み込む 09/gallery/practice/js/app.js

続いてサムネイル写真画像の表示を行います。このレッスンのapp.jsファイルをBracketsで開いて、以下のコードを記述してください。

まず、サムネイルを表示する枠の要素を取得して、thumbFlameという変数に格納しておきます❶。

次に繰り返し処理を使って、アルバムデータの読み込みとサムネイル写真画像を表示します❷。繰り返し処理の中では、まずサムネイルとなるimg要素を作成し、thumbImageという変数に格納します❸。

次に、表示する写真画像のファイル名をsrc属性にセットして読み込ませます。さらに、写真画像に関するキャプションを、alt属性にセットします❹。最後に、HTMLに反映するため、thumbFlameの子要素にthumbImageを追加しています❺。

ここまでできたら、コードを上書き保存して、一度ブラウザで確認してみましょう❻。

```
021
022 // サムネイル写真画像の表示
023 var thumbFlame = document.querySelector('#gallery .thumb');
024 for (var i = 0; i < album.length; i++) {
025   var thumbImage = document.createElement('img');
026   thumbImage.setAttribute('src', album[i].src);
027   thumbImage.setAttribute('alt', album[i].msg);
028   thumbFlame.insertBefore(thumbImage, null);
029 }
```

❶ 要素を取得
❷ 繰り返し処理
❸ img要素を作成
❹ 属性をセット
❺ 要素を追加

写真画像データがすべて表示された

この時点ではサムネイル画像も大きいままで表示されます。CSSを編集してサムネイルサイズに縮小します。

Lesson 49 [CSSの実践]

CSSで見た目を装飾しましょう

このレッスンのポイント

続いて、CSSで外観を整えていきます。CSSファイルにスタイルを記述しておけば、後からJavaScriptで追加したHTML要素にもスタイルが適用されます。ですからここではJavaScriptは特に編集せず、CSSファイルのみを編集していきます。

JavaScriptでのCSS操作は最小限にする

Chapter 6で説明したように、JavaScriptでHTML要素に直接スタイルを設定することもできますが、そうするとCSSファイルで指定したスタイルと別にJavaScriptで設定したスタイルが混在して管理しにくくなってしまいます。

一般的には、必要なスタイルをCSSファイルにあらかじめ記述しておいて、JavaScriptではclass属性やid属性を操作することで、適用するスタイルを切り替えます。これならCSSファイルだけでスタイルを一括管理できます。

▶ スタイルを適用するHTMLの構造

```
__<div_id="gallery">
____<div_class="main">
______<img_src="img/1.jpg"_alt="山道の緑が気持ちいい">
______<p>山道の緑が気持ちいい</p>
____</div>
____<div_class="thumb">
______<img_src="img/1.jpg"_alt="山道の緑が気持ちいい">
______<img_src="img/2.jpg"_alt="階段きつかった">
______<img_src="img/3.jpg"_alt="高尾山薬王院！">
______<img_src="img/4.jpg"_alt="帰りはロープウェイでスイスイ">
______<img_src="img/5.jpg"_alt="〆のお蕎麦です">
____</div>
__</div>
```

全体のスタイルを調整

メインの写真に枠を付ける

キャプションを目立たせる

サムネイルを丸く切り抜く

全体のスタイルを指定する

1 全体のスタイルを指定する

09/gallery/practice/css/style.css

まずは全体のスタイルを指定していきます。このレッスンのstyle.cssファイルをBracketsで開いて、以下のコードを記述してください。

まずはbody要素で基礎的な部分を調整します。写真が映えるよう背景色を「background-color: #444;」でダークグレーに指定し、要素のサイズ指定にボーダーの太さが含まれるように「box-sizing: border-box;」を指定します。また、サムネイル画像やキャプションが中央ぞろえになるように「text-align: center;」を指定します❶。

```
001 body_{
002 __background-color:_#444;
003 __box-sizing:_border-box;
004 __text-align:_center;
005 }
```

1 背景色を設定して中央ぞろえにする

2 フォトギャラリーのレイアウトを指定する

フォトギャラリー全体（id名がgalleryのdiv要素）のレイアウトを調整します。中央ぞろえにして、やや上部に余白をとるため「margin: auto;」と「padding-top: 40px;」を指定します。メイン画像の幅を一定にするために「width: 500px;」と指定します❶。

```
006
007 #gallery_{
008 __margin:_auto;
009 __padding-top:_40px;
010 __width:_500px;
011 }
```

2 余白を調整

CSSが苦手な人はこのまま写すだけでもOKですが、CSSの知識を身につけるとJavaScriptで表現できることもぐっと広がりますよ。

NEXT PAGE

写真とキャプションのスタイルを指定する

1 写真に枠を付ける 09/gallery/practice/css/style.css

ここでは、メインの写真（class属性がmainの要素の子のimg要素）に枠を付けていきます。
まず写真に白い枠線を付けるため「border: 4px solid #fff;」を指定しています。さらにシャドーで立体感を出すために「box-shadow: 0px 0px 14px #;」で黒いシャドーを付加しています。最後に、画像がギャラリーの幅いっぱいに広がるように「width: 100%;」を指定します1。

```
012
013 #gallery_.main_img_{
014 __border:_4px_solid_#fff;
015 __box-shadow:_0px_0px_14px_#000;
016 __width:_100%;
017 }
```

1 写真に枠を設定

2 キャプションを目立たせる

キャプション（class属性がmainの要素の子のp要素）が目立つように色を「color: #bbb;」で白色に、文字サイズを「font-size: 20px;」でやや大きく、文字の太さを「font-weight: bold;」で太めにします1。

```
018
019 #gallery_.main_p_{
020 __color:_#bbb;
021 __font-size:_20px;
022 __font-weight:_bold;
023 }
```

1 キャプションを設定

サムネイルのスタイルを指定する

1 画像を小さく、丸くする 09/gallery/practice/css/style.css

サムネイルの画像（class属性がthumbの要素の子のimg要素）を小さく、丸くしていきます。
まず、画像を丸くするために「border-radius: 400px;」を指定します。次に、小さな円形で切り抜くために、高さと幅を「height: 60px;」「width: 60px;」で指定してます。最後に、サムネイルの間に間隔を設けるため「margin: 10px;」で余白を指定します1。

```
024
025 #gallery_.thumb_img_{
026 __border-radius:_400px;
027 __height:_60px;
028 __margin:_10px;
029 __width:_60px;
030 }
```

1 サムネイルを整える

2 サムネイルに枠を付ける

最後に、メインの写真と同じように白枠とシャドーを指定して完成です1。

```
025 _#gallery_.thumb_img_{
026 __border:4px_solid_#fff;
027 __border-radius:_400px;
028 __box-shadow:_0px_0px_10px_#000;
029 __height:_60px;
030 __margin:_10px;
031 __width:_60px;
032 }
```

1 白枠とシャドーを設定

スタイルが適用された

Lesson 50 [イベント処理の実践]

表示する写真画像を選択できるようにしましょう

このレッスンのポイント

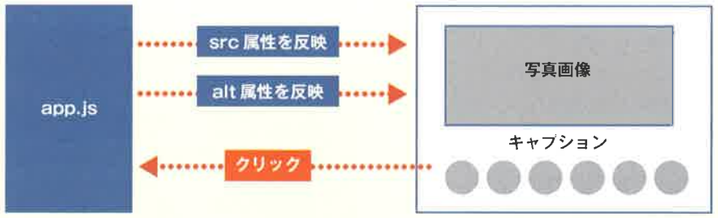
いよいよ最後の仕上げです。サムネイルをクリックしたら、メインに表示する写真が切り替わるようにしましょう。クリックで発生するイベントを利用して、メインの画像のsrc属性とalt属性を書き替えます。それだけでメイン画像の表示が更新されます。

クリックされた要素の情報を利用する

フォトギャラリーのメインとなる機能ですが、実現するのはそう難しくありません。

サムネイルがクリックされたことを検出するのは、Chapter 7で説明したイベントを利用すれば大丈夫です。サムネイル画像のsrc属性に画像のファイル名、alt属性にキャプションがセットされているので、それらをメインの画像とキャプションに移し替えれば写真が切り替わります。この処理は、イベントリスナー内でHTMLの属性値を取得・設定するメソッドを利用することで実現できます。

▶ メインの画像を切り替える仕組み



写真画像を選択できるようにする

1 クリックイベントを登録する 09/gallery/practice/js/app.js

このレッスンのapp.jsファイルをBracketsで開いて、以下のコードを記述してください。

まずはクリックイベントを登録します。サムネイルのimg要素すべてにクリックイベントを登録していくとキリがないので、サムネイルの親要素である「thumbFlame」にクリックイベントを登録して❶、イベントが発生したときにクリックされたのがimg要素かどうか判定するようにします❷。

```
030
031 //_クリックした画像をメインにする
032 thumbFlame.addEventListener('click',_function(event)_{
033 __if_(event.target.src)_{
034 ____//_ここに処理を記述していく
035 __}
036 });
```

1 クリックイベントに登録

2 img要素かどうかを確認

Point img要素かどうかを判定する

クリックされた要素は、引数として無名関数に渡されたeventオブジェクトのtargetプロパティを用いて調べることができます。img要素のオブジェクトは、src属性の値を確認できるsrcプロパティを持っているので、if文で対象のオブジェクトにsrcプロパティが存在することを調べ、その場合のみ表示処理を実行するようにします。

なお、HTMLの要素が持つすべての属性が、JavaScriptのオブジェクトにプロパティとして提供されているわけではありません。src属性など主要なもののみです。プロパティがないものについては、getAttributeメソッドやsetAttributeメソッドを使って操作します（P.136、P.181参照）。

NEXT PAGE

2 写真画像とキャプションを変更する

クリックした写真をメインの写真として表示する処理を記述します。「mainImage」のsrc属性に対し、クリックされたimg要素のsrc属性値を代入すれば、写真を表示することができます❶。同じ手順で「mainMsg」のテキストを、クリックされたimg要素のalt属性値に指定します❷。

```
//_クリックした画像をメインにする
thumbFlame.addEventListener('click',_function(event)_{
__if_(event.target.src)_{
____mainImage.src_=_event.target.src;
____mainMsg.innerText_=_event.target.alt;
__}
});
```

1 src属性をセット

2 alt属性をセット

3 フォトギャラリーが完成した

お疲れさまでした。これまでに記述したコードを上書き保存して、ブラウザでindex.htmlを開いて動作を確認してみましょう。写真が切り替われば完成です。

写真画像とキャプションが切り替わった

1 サムネイルをクリック

お疲れさまでした。これでJavaScriptの基本はバッチリです。ここから先はさらに一歩進んだJavaScriptの利用方法を学んでいきます。

Chapter 10

便利なjQueryを使用してみよう

この章では、JavaScriptをより便利に利用することができるjQueryという技術を学びます。Web制作の現場で標準的に使われているライブラリです。