

# 第7章

## JavaScriptを 利用したコンポーネント

前章までは、JavaScriptを意識することなく、データ属性を利用してコンポーネントに動きを追加する方法を紹介しました。たとえば、アラートコンポーネントに `data-dismiss="alert"` を指定するだけで、アラートを閉じる機能を追加できました。このような方法をデータ属性 API と言いますが、本章では、JavaScriptコードを記述して、コンポーネントにより柔軟な機能を追加する方法についても解説します。

## 7

## 1

SECTION

# Bootstrap の JavaScript プラグイン

Bootstrap の JavaScript プラグインを利用すると、コンポーネントに動きを追加することができます。プラグインは、Bootstrap が提供する個別の \*.js ファイルを必要に応じて組み込むか、一括版を使用すれば利用できます。

## 7.1.1 Bootstrap の JavaScript ファイル

Bootstrap の個別の JavaScript ファイルは、ソースファイルの js/dist/ 内にあります。またそれらを結合した一括版にも 2 種類あり、ドロップダウン、ポップオーバー、ツールチップで必要になる Popper.js が組み込まれている bootstrap.bundle.js (もしくはその圧縮版の bootstrap.bundle.min.js) と、Popper.js が含まれていない bootstrap.js (もしくはその軽量版の bootstrap.min.js) があります (表 7-1)。

▼表 7-1 Bootstrap の JavaScript ファイル

	JavaScript プラグイン	説明
bootstrap.bundle.js bootstrap.bundle.min.js	alert.js	アラートメッセージ
	button.js	コントロールボタン
	carousel.js	カルーセル
	collapse.js	折り畳み、ナビゲーションバーの切り替え
	dropdown.js	ドロップダウン
	modal.js	モーダル
	popover.js	ポップオーバー
	scrollspy.js	スクロールスパイ
	tab.js	タブ、リストグループ切り替えパネル
	tooltip.js	ツールチップ
	util.js	ユーティリティ機能。他の JavaScript ファイルと一緒に読み込む必要あり
	popper.js	ドロップダウン、ポップオーバー、ツールチップ使用時に必要

- 使わない機能を取り除いて、個別のプラグインを利用するとファイルサイズを小さくできますが、プラグインの依存関係に注意しましょう。ドロップダウン、ポップオーバー、ツールチップは popper.js に依存していて、util.js はすべてのプラグインに必要です。
- また、Bootstrap の JavaScript のコードは、jQuery を利用しているため、jQuery を最初に読み込む必要があります。たとえばドロップダウンを使用する場合、jQuery、popper.js、util.js、alert.js の順で記述します (リスト 7-1)。

## ▼リスト 7-1 ドロップダウンプラグインの例

```
<script src="js/jquery-3.3.1.slim.min.js"></script><!-- jQuery -->
<script src="js/popper.js"></script><!-- ドロップダウンプラグインに必要 -->
<script src="js/util.js"></script><!-- 全プラグインに必要 -->
<script src="js/dropdown.js"></script><!-- ドロップダウンプラグイン -->
```

本書のサンプルはすべてのプラグインを使用可能にするために、全部入りの軽量版 bootstrap.bundle.min.js を前提として作成しています（リスト 7-2）。

## ▼リスト 7-2 すべてのプラグインを使用可能

```
<script src="js/jquery-3.3.1.slim.min.js"></script><!-- jQuery -->
<script src="js/bootstrap.bundle.min.js"></script><!-- 全部入り -->
```

## 7.1.2 Bootstrap のデータ属性 API

ほとんどの Bootstrap プラグインは、HTML にデータ属性を追加するだけで利用できます。たとえば、アラートコンポーネントに data-dismiss="alert" を指定して、アラートを閉じる機能を追加しました（P.113 参照）。このような方法を **データ属性 API** と言いますが、時には、この機能を無効にしたい場合もあります。データ属性 API を無効にするには、jQuery の off メソッドを使います（リスト 7-3）。

## ▼リスト 7-3 データ属性 API を無効にする

```
$(document).off('.data-api');
```

特定のプラグインを対象とするには、data-api の名前空間とともにプラグイン名を指定します。たとえばリスト 7-4 のようにすると、アラートコンポーネントのデータ属性 API を無効にします。

## ▼リスト 7-4 アラートのデータ属性 API を無効にする

```
$(document).off('.alert.data-api');
```

## 2

## カルーセル

コンポーネントで JavaScript を利用する際の基本をおさえたところで、カルーセルコンポーネントの使い方を見ていきましょう。

Bootstrap のカルーセルは、画像やテキストなどのコンテンツを、横方向に循環させるスライドショーにするためのコンポーネントです。カルーセルを使用することで、Web サイトのメインビジュアルなどのスライドショーを簡単に実装できます。また、必要に応じて、スライドを前後に送るコントローラーや、スライド数を表示・カウントするインジケーター、スライドのキャプションを表示することもできます。

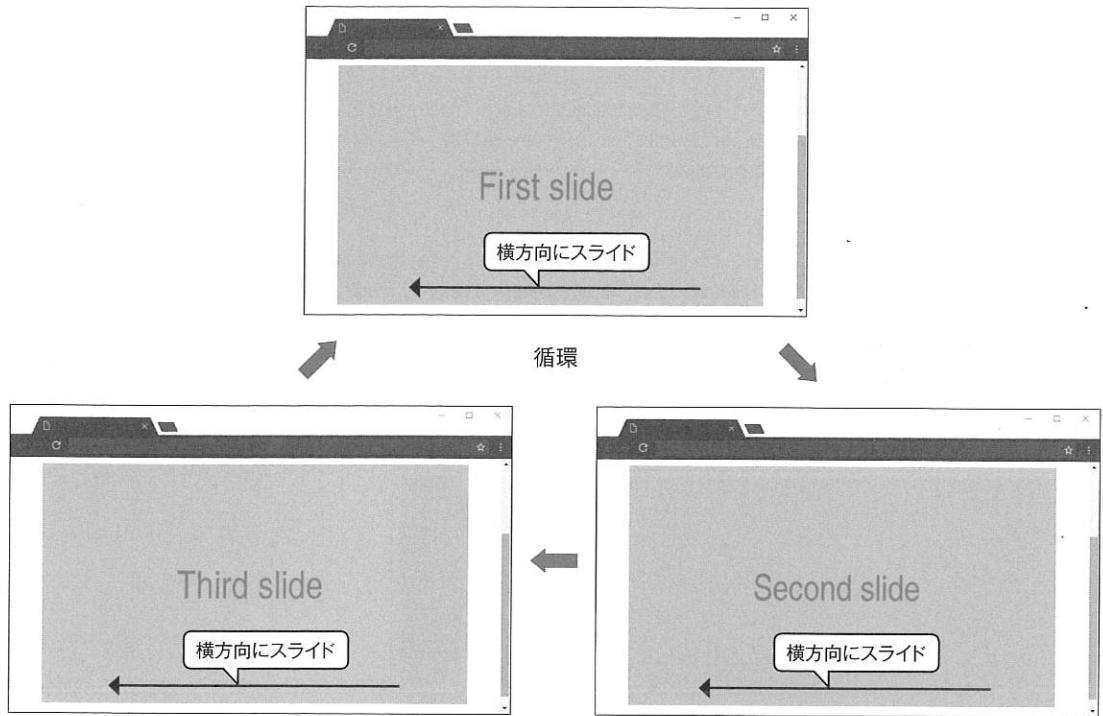
### 7.2.1 基本的な使用例

まずは 3 枚のスライドを循環させる基本的なカルーセルの作り方を見ていきましょう（リスト 7-5、図 7-1）。

▼リスト 7-5 カルーセルの基本的な使用例（carousel-basic.html）

```
<div class="carousel slide" data-ride="carousel"> _____①
  <div class="carousel-inner"> _____②
    <!-- First slide -->
    <div class="carousel-item active"> _____③
       _____④
    </div>
    <!-- Second slide -->
    <div class="carousel-item"> _____③
       _____④
    <!-- Third slide -->
    <div class="carousel-item"> _____③
       _____④
    </div>
  </div>
</div>
```

▼図 7-1 カルーセルの基本的な使用例



カルーセルを作成する場合は、まずカルーセルの外枠となる要素に **carousel** クラスと **slide** クラス、属性 **data-ride="carousel"** を追加します（①）。

カルーセルの内枠として複数のスライドをまとめる要素には **carousel-inner** クラスを追加します（②）。各スライドには **carousel-item** クラスを追加し、初期画面で表示されるスライドには **active** クラスを加えます（③）。スライドの中に active クラスがない場合は、カルーセルが表示されなくなるので注意してください。

なお、このコンポーネントはスライド内のコンテンツのサイズを自動調整してくれません。この例では、カルーセル内の画像が適切なサイズに表示されるように、各スライド画像に Display ユーティリティ（P.310 参照）の **d-block** クラス、Sizing ユーティリティ（P.314 参照）の **w-100** クラスを追加しています（④）。

## 7.2.2 コントローラーを表示させる

次の例では、カルーセルに「<」（前に戻るアイコン）や、「>」（次へ送るアイコン）といった、スライドを前後に送るコントローラーを表示させています（リスト 7-6、図 7-2）。

▼リスト 7-6 コントローラー付きカルーセル（carousel-controls.html）

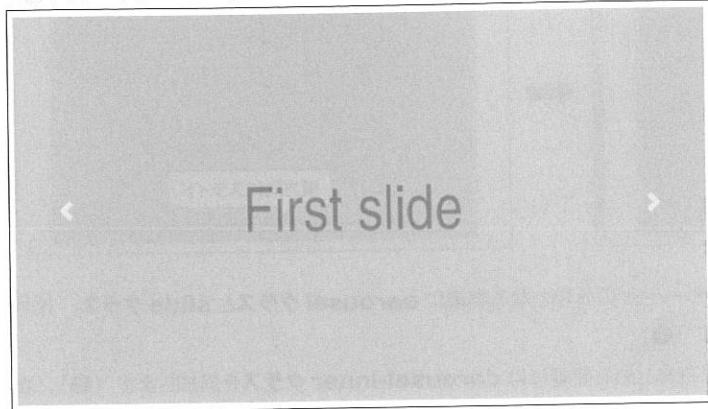
```
<div id="carouselControls" class="carousel slide" data-ride="carousel"> _____ ①
  <!-- カルーセル部分 -->
  <div class="carousel-inner">
```

```

...中略...
</div>
<!-- コントローラー -->
<!-- 前に戻るアイコン部分 -->
<a class="carousel-control-prev" href="#carouselControls" role="button" data-slide="prev"> ━━━━⑤
  <span class="carousel-control-prev-icon" aria-hidden="true"></span> ━━━━⑥
  <span class="sr-only">前に戻る</span> ━━━━⑦
</a>
<!-- 次に送るアイコン -->
<a class="carousel-control-next" href="#carouselControls" role="button" data-slide="next"> ━━━━②
  <span class="carousel-control-next-icon" aria-hidden="true"></span> ━━━━③
  <span class="sr-only">次に送る</span> ━━━━④
</a>
</div>

```

▼図 7-2 コントローラー付きカルーセル



カルーセルにコントローラーを表示する場合は、`carousel-inner` クラスが設定されたカルーセル内枠の後に、2つの `a` 要素を配置します。このとき、`carousel` クラスを指定したカルーセル外枠の ID と、コントローラーの `a` 要素の `href` 属性の値とを一致させ、コントローラーを有効化します（①）。

**次へ送る**コントローラーを作成する場合は、まず `a` 要素に **carousel-control-next** クラスを追加し、属性 `data-slide="next"` を加えて JavaScript 経由でスライドを次に送る機能を有効化します（②）。次に `span` 要素に **carousel-control-next-icon** クラスを追加し、「>」アイコンを作成します（③）。またスクリーンリーダーなどの支援技術に対してアイコンの意味を伝えるために、`span` 要素にスクリーンリーダー用ユーティリティ（P.354 参照）の **sr-only** クラスを追加して「次に送る」などの非表示テキストを加えておきましょう（④）。

**前に戻る**コントローラーを作成する場合は、同じように、`a` 要素に **carousel-control-prev** クラスを追加し、属性 `data-slide="prev"` を加えて JavaScript 経由でスライドを前に戻す機能を有効化します（⑤）。次に `span` 要素に **carousel-control-prev-icon** クラスを追加し、「<」アイコンを作成します（⑥）。次へ送るコントローラーと同様、`span` 要素に **sr-only** クラスを追加して「次に送る」などの非表示テキストを加えておきましょう（⑦）。

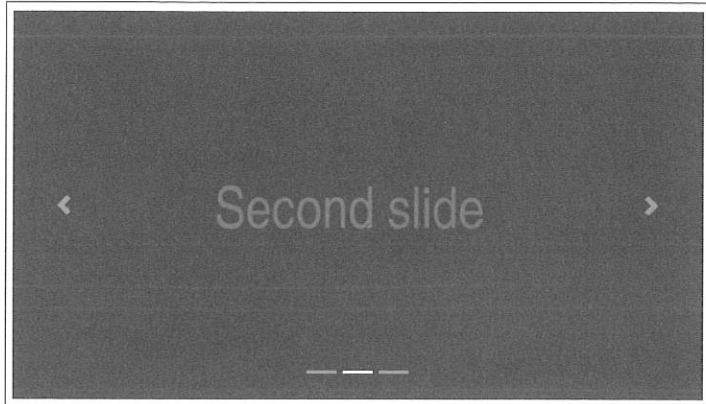
### 7.2.3 インジケーターを表示させる

次の例では、カルーセルにスライド数をカウントするインジケーターを表示させています（リスト7-7、図7-3）。

▼リスト7-7 インジケーター付きカルーセル（carousel-indicators.html）

```
<div id="carouselIndicators" class="carousel slide" data-ride="carousel">
  <!-- インジケーター部分 -->
  <ol class="carousel-indicators"> _____ ①
    <li data-target="#carouselIndicators" data-slide-to="0" class="active"></li> _____ ②
    <li data-target="#carouselIndicators" data-slide-to="1"></li>
    <li data-target="#carouselIndicators" data-slide-to="2"></li>
  </ol>
  <!-- カルーセル部分 -->
  <div class="carousel-inner">
    ...中略...
  </div>
  <!-- コントローラー部分 -->
  ...中略...
</div>
```

▼図7-3 インジケーター付きカルーセル



カルーセルにインジケーターを表示する場合は、carousel-innerクラスが設定されたカルーセル内枠の前に、**carousel-indicators** クラスを設定したol要素を配置します（①）。このとき、carouselクラスを設定したカルーセル外枠のIDと、インジケーター内のli要素の**data-target**属性の値を一致させ、属性**data-slide-to={スライド番号}**を追加します（②）。スライド番号には0からはじまる数値が入ります。また、初期画面で表示されるスライドに対応するli要素には**active**クラスを加えます。

### 7.2.4 スライドのキャプションを表示させる

カルーセルの各スライドにキャプションを表示させる場合、carousel-itemクラスが設定された各スライドの子

要素に、**carousel-caption** クラスを追加します。

次の例では、画面幅 md 以上でキャプションが表示されるように、Display ユーティリティ（P.310 参照）の **d-none** クラスと **d-md-block** を追加しています（リスト 7-8、図 7-4）。

▼リスト 7-8 キャプション付きカルーセル（carousel-captions.html）

```
<!-- slide -->
<div class="carousel-item">
  
  <!-- キャプション -->
  <div class="carousel-caption d-none d-md-block">
    <h5>スライド見出し</h5>
    <p>スライドのキャプション文</p>
  </div>
</div>
```

▼図 7-4 キャプション付きカルーセル



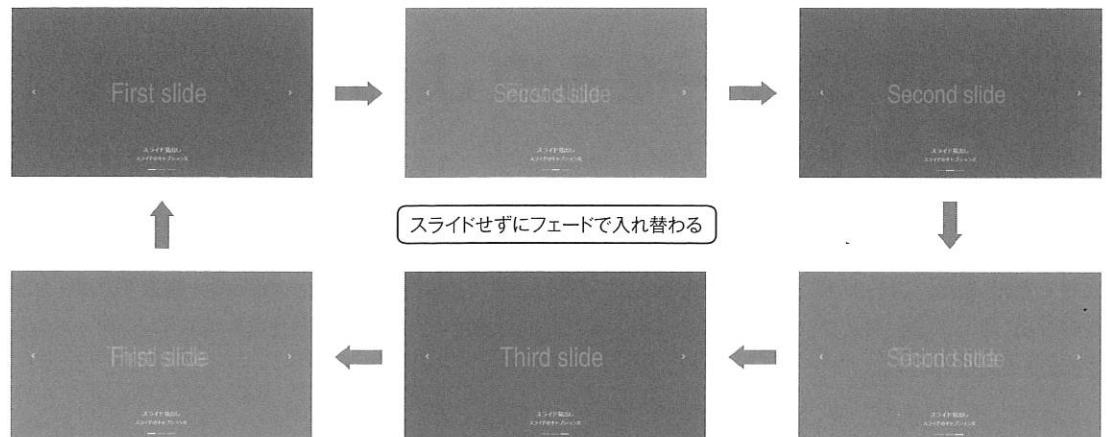
### 7.2.5 フェードで遷移させる 4.1

カルーセルに **carousel-fade** クラスを追加すると、スライドではなくフェードで遷移させることができます（リスト 7-9、図 7-5）。

▼リスト 7-9 フェードで遷移するカルーセル（carousel-crossfade.html）

```
<div id="carouselSample" class="carousel slide carousel-fade" data-ride="carousel">
```

▼図 7-5 フェードで遷移させる



## NOTE

## 遷移期間を変更するには?

カルーセルの遷移スピードを変更するには、カスタムスタイルで、carousel-item クラスの transition プロパティの 2 つ目の値（デフォルトは .6s）を変更します。

## ▼リスト 7-10 カスタムスタイルで遷移期間を変更する

```
.carousel-item {  
    transition: transform .2s ease; // 遷移スピードを2ミリ秒に変更  
}
```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10

7

3

## カルーセルの JavaScript 使用

カルーセルコンポーネント（P.262 参照）は、**data-ride="carousel"** 属性を指定することで使用することができましたが、次のように、JavaScript 経由で呼び出すこともできます。JavaScript 経由でカルーセルを呼び出したり、後述のオプションを指定する場合は、**data-ride="carousel"** 属性は使用できないので注意しましょう（リスト 7-11）。

▼リスト 7-11 カルーセルを JavaScript 経由で呼び出す（carousel-js.html）

```
<div id="carouselExample" class="carousel slide"><!-- data-ride="carousel"は不要 -->
...中略...
<script>
$(function(){
  $('#carouselExample').carousel();
});
</script>
```

カルーセルで定義されているオプションは表 7-2 のとおりです。

▼表 7-2 カルーセルのオプション

名前	デフォルト	説明
interval	5000	自動的に切り替える時間（ミリ秒）。false の場合、カルーセルは自動的に切り替わらない
keyboard	true	カルーセルをキーボードイベントに反応させるかどうか
pause	"hover"	"hover" の場合、mouseenter でカルーセルを一時停止
ride	false	ユーザーが最初の項目を手動で動かした後、自動再生。"carousel" なら読み込み時に自動再生
wrap	true	最後まで再生した後、最初から繰り返すか、停止するかどうか

カルーセルのオプションは、データ属性または JavaScript を使用して渡せます。データ属性を使用する場合、**data-interval="1000"** のように、**data-** にオプション名を追加します（リスト 7-12）。

▼リスト 7-12 データ属性でオプションを指定（carousel-options-data.html）

```
<div id="carouselExample" class="carousel slide" data-ride="carousel" data-interval="1000">
```

JavaScript でオプションを渡す場合は次のように指定します（リスト 7-13）。

## ▼リスト 7-13 JavaScriptコードでオプションを指定 (carousel-options.js.html)

```
<div id="carouselExample" class="carousel slide"><!-- data-ride="carousel"は不要 -->
...中略...
<script>
$(function(){
    $('#carouselExample').carousel({
        interval: 1000
    });
});
</script>
```

**7.3.1 カルーセルのメソッド**

カルーセルで定義されているメソッドは表 7-3 のとおりです。

▼表 7-3 カルーセルのメソッド

メソッド	説明
.carousel('pause')	カルーセルの再生を停止
.carousel(number)	カルーセルを特定フレームに移動する (0 から数える)
.carousel('prev')	前のアイテムに移動
.carousel('next')	次のアイテムに移動
.carousel('dispose')	要素のカルーセルを破棄

次のコードは、メソッドを利用して各種コントロールボタンを作成した例です（リスト 7-14、図 7-6）。

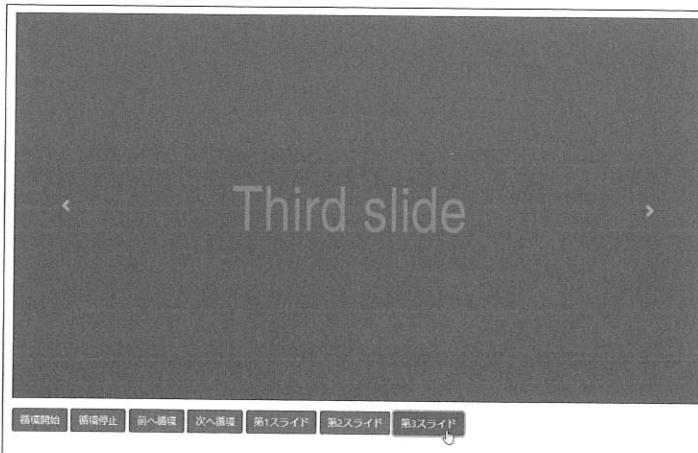
## ▼リスト 7-14 カルーセルメソッドの例 (carousel-methods.html)

```
<!-- コントロールボタン -->
<div class="control-buttons my-3">
    <input type="button" class="btn btn-primary start-slide" value="循環開始">
    <input type="button" class="btn btn-primary pause-slide" value="循環停止">
    <input type="button" class="btn btn-primary prev-slide" value="前へ循環">
    <input type="button" class="btn btn-primary next-slide" value="次へ循環">
    <input type="button" class="btn btn-primary slide-first" value="第1スライド">
    <input type="button" class="btn btn-primary slide-second" value="第2スライド">
    <input type="button" class="btn btn-primary slide-third" value="第3スライド">
</div>
...中略...
<script>
$(function(){
    // 循環開始
    $(".start-slide").on('click',function(){
        $("#carousel").carousel('cycle');
    });
    // 一時停止
```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10

```
$(".pause-slide").on('click',function(){
    $("#carousel").carousel('pause');
});
// 前へ循環
$(".prev-slide").on('click',function(){
    $("#carousel").carousel('prev');
});
// 次へ循環
$(".next-slide").on('click',function(){
    $("#carousel").carousel('next');
});
// 特定のフレームに循環
$(".slide-first").on('click',function(){
    $("#carousel").carousel(0);
});
$(".slide-second").on('click',function(){
    $("#carousel").carousel(1);
});
$(".slide-third").on('click',function(){
    $("#carousel").carousel(2);
});
});
</script>
```

▼図 7-6 カルーセルのメソッド使用例



### 7.3.2 カルーセルのイベント

カルーセルで定義されているイベントは表 7-4 のとおりです。カルーセルのスライド遷移前や後に何らかの処理を挟みたい場合に使用できます。

▼表 7-4 カルーセルのイベント

イベント	説明
slide.bs.carousel	カルーセルのスライドが遷移する直前に発動
slid.bs.carousel	カルーセルのスライドが遷移した直後に発動

次のサンプルでは、キャプションをフワっと表示させるエフェクトを追加するために、スライド遷移前にキャプションを非表示にし、遷移後に表示しています（リスト 7-15）。

▼リスト 7-15 カルーセルのイベント : slid.bs.carousel (carousel-events.html)

```
<script>
$('#carouselExample').on('slide.bs.carousel', function () {
  $('#carouselExample .carousel-caption').hide();
});
$('#carouselExample').on('slid.bs.carousel', function () {
  $('#carouselExample .carousel-caption').show();
});
</script>
```

NOTE

**よりフワっと表示させるには？**

サンプルでは、jQuery のスリムバージョン（jquery-XXX.slim.min.js）を使用しているため、show() を利用していますが、jQuery の通常版（jquery-XXX.min.js）を利用すれば fadeIn() を利用して、よりフワとしたフェードイン効果を追加することができます（リスト 7-16）。

▼リスト 7-16 fadeIn() を利用してキャプションをフェードする

```
$('#carouselExample').on('slid.bs.carousel', function () {
  $('#carouselExample .carousel-caption').fadeIn();
});
```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10

## 4

## 折り畳み

Bootstrap の折り畳みは、コンテンツの表示と非表示とを切り替える開閉パネルなどを作成するコンポーネントです。Bootstrap の定義済みクラスと、JavaScript のプラグインを使用して、簡単に実装することができます。

## 7.4.1 基本的な使用例

折り畳みは、切り替えボタンをクリックして、切り替え対象となる要素のクラスを変更し、コンテンツの表示と非表示とを切り替えます。表示の切り替えに使用される定義済みクラスを表 7-5 のとおりです。

▼表 7-5 折り畳みで主に使用される定義済みクラス

クラス	概要
collapse	コンテンツを非表示にする
collapsing	時間的変化を伴ってコンテンツを非表示にする（遷移が開始すると自動的に追加され、終了すると削除される）
collapse show	コンテンツを表示する

次の例では、基本的な折り畳みを作成しています（リスト 7-17、図 7-7、図 7-8）。

▼リスト 7-17 折り畳みの基本的な使用例（collapse-basic.html）

```
<!-- 切り替えボタン -->
<p>
<!-- a要素とhref属性による切り替えボタン -->
<p>
  <a class="btn btn-secondary" data-toggle="collapse" href="#collapseContent01" role="button" aria-expanded="false" aria-controls="collapseContent01">a要素とhref属性によるボタン</a> ①
</p>
<!-- 切り替える対象となるコンテンツ -->
<div class="collapse" id="collapseContent01"> ③
  <div class="card card-body">
    a要素の切り替えボタンをクリックすることで表示と非表示とが切り替わるコンテンツ
  </div>
</div>

<!-- button要素とdata-target属性による切り替えボタン -->
<p>
  <button class="btn btn-secondary" type="button" data-toggle="collapse" data-target="#collapseContent02" aria-expanded="false" aria-controls="collapseContent02"> ②
  #collapseContent02
  </button>
</p>
```

```

button要素とdata-target属性によるボタン</button> _____ ②
</p>
<!-- 切り替える対象となるコンテンツ -->
<div class="collapse" id="collapseContent02"> _____ ③
  <div class="card card-body">
    button要素の切り替えボタンをクリックすることで表示と非表示とが切り替わるコンテンツ
  </div>
</div>

```

▼図 7-7 a 要素と href 属性による切り替えボタン



▼図 7-8 button 要素と data-target 属性による切り替えボタン



1  
2  
3  
4  
5  
6  
7  
8  
9  
10

まず切り替えボタンを作成します。切り替えボタンには、a要素またはbutton要素を使用します。a要素を使用する場合は、href属性の値に切り替え対象の要素のIDやクラスを指定します（①）。また、アクセシビリティへの配慮として属性role="button"を追加し、このa要素の役割がボタンであることをスクリーンリーダーなどの支援技術に伝えましょう。button要素を使用する場合は、data-target属性の値に、切り替え対象の要素のIDやクラスを指定します（②）。いずれの場合も、切り替えボタンには属性data-toggle="collapse"を追加し、スクリーンリーダーなどの支援技術に対して要素の状態を伝えるaria-\*属性を追加します。あらかじめ表示される要素には属性aria-expanded="true"を、非表示の要素には属性aria-expanded="false"を追加します。また属性aria-controls="（切り替える対象となる要素のIDやクラス）"を追加し、切り替える対象であることをスクリーンリーダーなどの支援技術に伝えましょう。

次に、切り替え対象の要素にcollapseを追加し、切り替えボタンによって表示と非表示とが切り替わるコンテンツ部分を作成します（③）。

## 7.4.2 複数の要素の表示と非表示とを切り替える

複数のa要素のhref属性、またはbutton要素のdata-target属性に、それぞれ別の値を設定することで、複数の要素の表示と非表示とを切り替えることができます。

次の例では、切り替え対象の要素のIDをdata-target属性の値に指定することで個別コンテンツの表示を切り替え、共通のクラスを指定することで複数コンテンツの表示を同時に切り替える折り畳みを作成しています（リスト7-18、図7-9）。

▼リスト7-18 複数の要素の表示と非表示とを切り替える（collapse-multiple.html）

```
<p>
    <!-- ID「content-01」の切り替えボタン -->
    <button class="btn btn-secondary" type="button" data-toggle="collapse" data-target="#content-01" ↪
        aria-expanded="false" aria-controls="content-01">ID「content-01」を表示切り替え</button>
    <!-- ID「content-02」の切り替えボタン -->
    <button class="btn btn-secondary" type="button" data-toggle="collapse" data-target="#content-02" ↪
        aria-expanded="false" aria-controls="content-02">ID「content-02」を表示切り替え</button>
    <!-- クラス「contents」の切り替えボタン -->
    <button class="btn btn-secondary" type="button" data-toggle="collapse" data-target=".contents" ↪
        aria-expanded="false" aria-controls="contents-01 contents-02">クラス「contents」を同時に表示切り替え</button>
</p>
<div class="row">
    <div class="col">
        <!-- ID「content-01」、クラス「contents」 -->
        <div class="collapse contents" id="content-01">
            <div class="card card-body">
                ID「content-01」、class「contents」
            </div>
        </div>
    </div>
</div>
```

```
<div class="col">
  <!-- ID「content-02」、クラス「contents」-->
  <div class="collapse contents" id="content-02">
    <div class="card card-body">
      ID「content-02」、クラス「contents」
    </div>
  </div>
</div>
```

▼図 7-9 複数の要素の表示と非表示とを切り替える



### 7.4.3 アコーディオンを作成する

アコーディオンとは、コンテンツのヘッダー部分をクリックすることで、コンテンツの表示と非表示を折り畳みによって切り替える機能です。Bootstrapでは、折り畳みとカード（P.124 参照）のコンポーネントを組み合わせることで、アコーディオンを作成することができます。

#### ■ 基本的なアコーディオン

次の例では、3つのカード内にヘッダーとコンテンツを配置し、折り畳みを利用してコンテンツの表示を切り替えるアコーディオンを作成しています。カードの見出しの切り替えボタンをクリックすると、コンテンツの表示・非表示が切り替えられます。また、他のカードの切り替えボタンをクリックするとコンテンツが非表示になります（リスト7-19、図7-10）。カードの作成については「カード」（P.124）を参照してください。

## ▼リスト7-19 基本的なアコーディオン (collapse-accordion.html)

```

<div class="accordion" id="accordion"> _____ ①
  <!-- カード01 -->
  <div class="card">
    <!-- カードヘッダー -->
    <div class="card-header" id="headingOne">
      <h5 class="mb-0">
        <button class="btn btn-link" type="button" data-toggle="collapse" data-target="#collapseOne" aria-expanded="true" aria-controls="collapseOne"> _____ ②
          カード01の切り替えボタン
        </button>
      </h5>
    </div>
    <!-- コンテンツ -->
    <div id="collapseOne" class="collapse show" aria-labelledby="headingOne" data-parent="#accordion"> _____ ③
      <div class="card-body">
        ...中略...
      </div>
    </div>
    </div>
  <!-- カード02 -->
  <div class="card">
    <!-- カードヘッダー -->
    <div class="card-header" id="headingTwo">
      <h5 class="mb-0">
        <button class="btn btn-link collapsed" type="button" data-toggle="collapse" data-target="#collapseTwo" aria-expanded="false" aria-controls="collapseTwo"> _____ ②
          カード02の切り替えボタン
        </button>
      </h5>
    </div>
    <!-- コンテンツ -->
    <div id="collapseTwo" class="collapse" aria-labelledby="headingTwo" data-parent="#accordion"> _____ ③
      <div class="card-body">
        ...中略...
      </div>
    </div>
    </div>
  <!-- カード03 -->
  <div class="card">
    <!-- カードヘッダー -->
    <div class="card-header" id="headingThree">
      <h5 class="mb-0">
        <button class="btn btn-link collapsed" type="button" data-toggle="collapse" data-target="#collapseThree" aria-expanded="false" aria-controls="collapseThree"> _____ ②
          カード03の切り替えボタン
        </button>
      </h5>
    </div>
  </div>

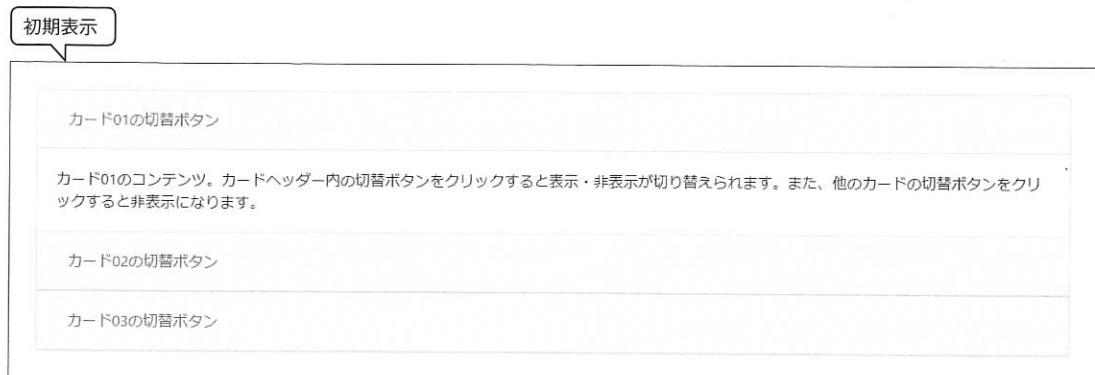
```

```

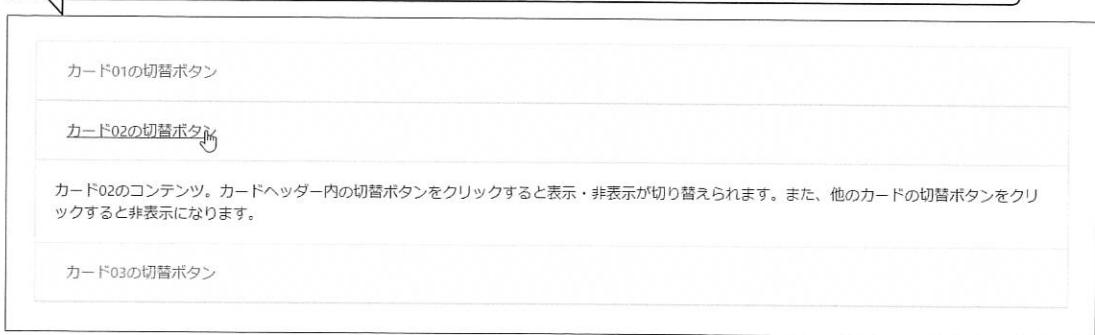
</h5>
</div>
<!-- コンテンツ -->
<div id="collapseThree" class="collapse" aria-labelledby="headingThree" data-parent="#accordion">③
  <div class="card-body">
    ...中略...
  </div>
</div>
</div>
</div>

```

▼図 7-10 基本的なアコーディオン



他のカードの切り替えボタンをクリックすると、そのカードのコンテンツが表示され、開いていたコンテンツは非表示になる



まず、アコーディオンの外枠を作成します。アコーディオンの外枠は、複数カードを含む親要素に **accordion クラス** を追加して作成します（①）。

次に、各カードヘッダーの見出し内に折り畳みの切り替えボタンを作成します。この例では、button 要素に属性 **data-toggle="collapse"** を追加します。さらに **data-target 属性** を追加し、値に切り替え対象の要素の ID を指定しています（②）。

そして、切り替えの対象となるコンテンツを作成します。コンテンツ部分は、card-body クラスが設定されたカード本文の親要素に **collapse クラス** を追加し、**data-parent 属性** の値にアコーディオンの外枠の要素の

IDを指定します（③）。このとき、初期表示されるコンテンツには**collapseクラス**が設定された要素に**showクラス**を追加します。さらに、aria-labelledby属性の値にカードヘッダーのIDを指定し、スクリーンリーダーなどの支援技術に対してカードヘッダーがこのコンテンツのラベルとして関連付けられていることを伝えましょう。

### COLUMN WAI-ARIAとは？

Bootstrapのコンポーネントで度々出てくるaria属性やrole属性は、W3Cによって定められているWAI-ARIA (<https://www.w3.org/TR/wai-aria-1.1/>) という仕様の一部です。WAI-ARIAは、「Web Accessibility Initiative - Accessible Rich Internet Applications」の頭文字をとったもので、HTMLやSVGで利用できるアクセシビリティ向上のための属性の仕様です。

この仕様では、主にrole属性とaria属性の2つが定義されています。role属性は、コンテンツの役割を定義します。例えば、role="navigation"やrole="banner"、role="search"など、要素が何なのか、何をするものなのかを定義します。

aria属性は、コンテンツの状態を表したり、性質を表すために使用します。例えば、aria-disabled="true"は、フォームのinputが現在disabledの状態であることをスクリーンリーダーに対して伝えます。また別の例で、aria-required="true"は、フォームのinputに値を入力しなければならない性質を表します。

Bootstrapの対話式コンポーネントは、タッチ、マウス、キーボード等のユーザーに対応しています。関連するWAI-ARIAのrole属性を使用することで、スクリーンリーダーなどの支援技術でも、これらのコンポーネントの役割を理解でき、操作可能になることを目指しています。

## ▼リスト 7-13 JavaScriptコードでオプションを指定 (carousel-options.js.html)

```
<div id="carouselExample" class="carousel slide"><!-- data-ride="carousel"は不要 -->
...中略...
<script>
$(function(){
    $('#carouselExample').carousel({
        interval: 1000
    });
});
</script>
```

**7.3.1 カルーセルのメソッド**

カルーセルで定義されているメソッドは表 7-3 のとおりです。

▼表 7-3 カルーセルのメソッド

メソッド	説明
.carousel('pause')	カルーセルの再生を停止
.carousel(number)	カルーセルを特定フレームに移動する (0 から数える)
.carousel('prev')	前のアイテムに移動
.carousel('next')	次のアイテムに移動
.carousel('dispose')	要素のカルーセルを破棄

次のコードは、メソッドを利用して各種コントロールボタンを作成した例です（リスト 7-14、図 7-6）。

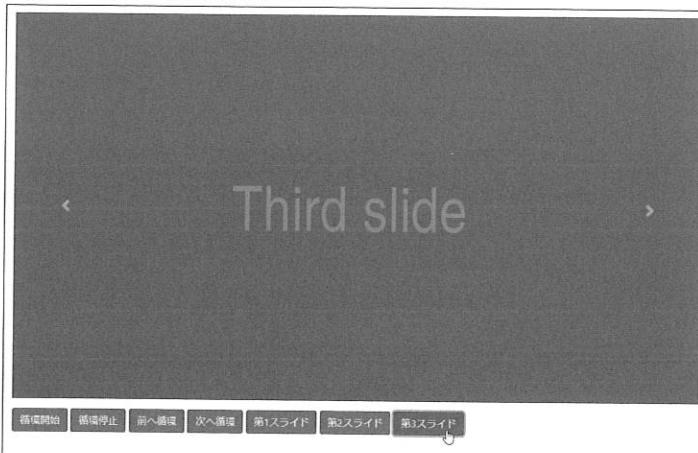
## ▼リスト 7-14 カルーセルメソッドの例 (carousel-methods.html)

```
<!-- コントロールボタン -->
<div class="control-buttons my-3">
    <input type="button" class="btn btn-primary start-slide" value="循環開始">
    <input type="button" class="btn btn-primary pause-slide" value="循環停止">
    <input type="button" class="btn btn-primary prev-slide" value="前へ循環">
    <input type="button" class="btn btn-primary next-slide" value="次へ循環">
    <input type="button" class="btn btn-primary slide-first" value="第1スライド">
    <input type="button" class="btn btn-primary slide-second" value="第2スライド">
    <input type="button" class="btn btn-primary slide-third" value="第3スライド">
</div>
...中略...
<script>
$(function(){
    // 循環開始
    $(".start-slide").on('click',function(){
        $("#carousel").carousel('cycle');
    });
    // 一時停止
```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10

```
$(".pause-slide").on('click',function(){
    $("#carousel").carousel('pause');
});
// 前へ循環
$(".prev-slide").on('click',function(){
    $("#carousel").carousel('prev');
});
// 次へ循環
$(".next-slide").on('click',function(){
    $("#carousel").carousel('next');
});
// 特定のフレームに循環
$(".slide-first").on('click',function(){
    $("#carousel").carousel(0);
});
$(".slide-second").on('click',function(){
    $("#carousel").carousel(1);
});
$(".slide-third").on('click',function(){
    $("#carousel").carousel(2);
});
});
</script>
```

▼図 7-6 カルーセルのメソッド使用例



### 7.3.2 カルーセルのイベント

カルーセルで定義されているイベントは表 7-4 のとおりです。カルーセルのスライド遷移前や後に何らかの処理を挟みたい場合に使用できます。

▼表 7-4 カルーセルのイベント

イベント	説明
slide.bs.carousel	カルーセルのスライドが遷移する直前に発動
slid.bs.carousel	カルーセルのスライドが遷移した直後に発動

次のサンプルでは、キャプションをフワっと表示させるエフェクトを追加するために、スライド遷移前にキャプションを非表示にし、遷移後に表示しています（リスト 7-15）。

▼リスト 7-15 カルーセルのイベント : slid.bs.carousel (carousel-events.html)

```
<script>
$('#carouselExample').on('slide.bs.carousel', function () {
  $('#carouselExample .carousel-caption').hide();
});
$('#carouselExample').on('slid.bs.carousel', function () {
  $('#carouselExample .carousel-caption').show();
});
</script>
```

NOTE

**よりフワっと表示させるには？**

サンプルでは、jQuery のスリムバージョン（jquery-XXX.slim.min.js）を使用しているため、show() を利用していますが、jQuery の通常版（jquery-XXX.min.js）を利用すれば fadeIn() を利用して、よりフワとしたフェードイン効果を追加することができます（リスト 7-16）。

▼リスト 7-16 fadeIn() を利用してキャプションをフェードする

```
$('#carouselExample').on('slid.bs.carousel', function () {
  $('#carouselExample .carousel-caption').fadeIn();
});
```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10

## 4

## 折り畳み

Bootstrap の折り畳みは、コンテンツの表示と非表示とを切り替える開閉パネルなどを作成するコンポーネントです。Bootstrap の定義済みクラスと、JavaScript のプラグインを使用して、簡単に実装することができます。

## 7.4.1 基本的な使用例

折り畳みは、切り替えボタンをクリックして、切り替え対象となる要素のクラスを変更し、コンテンツの表示と非表示とを切り替えます。表示の切り替えに使用される定義済みクラスを表 7-5 のとおりです。

▼表 7-5 折り畳みで主に使用される定義済みクラス

クラス	概要
collapse	コンテンツを非表示にする
collapsing	時間的変化を伴ってコンテンツを非表示にする（遷移が開始すると自動的に追加され、終了すると削除される）
collapse show	コンテンツを表示する

次の例では、基本的な折り畳みを作成しています（リスト 7-17、図 7-7、図 7-8）。

▼リスト 7-17 折り畳みの基本的な使用例（collapse-basic.html）

```
<!-- 切り替えボタン -->
<p>
<!-- a要素とhref属性による切り替えボタン -->
<p>
  <a class="btn btn-secondary" data-toggle="collapse" href="#collapseContent01" role="button" aria-expanded="false" aria-controls="collapseContent01">a要素とhref属性によるボタン</a> ①
</p>
<!-- 切り替える対象となるコンテンツ -->
<div class="collapse" id="collapseContent01"> ③
  <div class="card card-body">
    a要素の切り替えボタンをクリックすることで表示と非表示とが切り替わるコンテンツ
  </div>
</div>

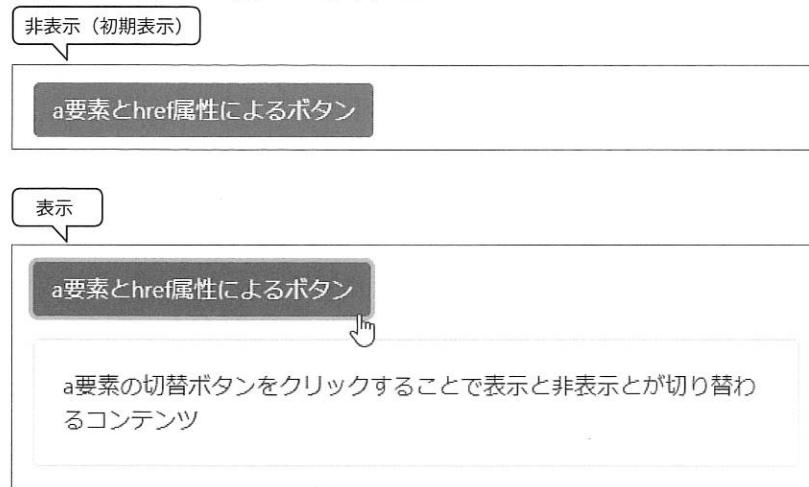
<!-- button要素とdata-target属性による切り替えボタン -->
<p>
  <button class="btn btn-secondary" type="button" data-toggle="collapse" data-target="#collapseContent02" aria-expanded="false" aria-controls="collapseContent02"> ②
  #collapseContent02
  </button>
</p>
```

```

button要素とdata-target属性によるボタン</button> _____ ②
</p>
<!-- 切り替える対象となるコンテンツ -->
<div class="collapse" id="collapseContent02"> _____ ③
  <div class="card card-body">
    button要素の切り替えボタンをクリックすることで表示と非表示とが切り替わるコンテンツ
  </div>
</div>

```

▼図 7-7 a 要素と href 属性による切り替えボタン



▼図 7-8 button 要素と data-target 属性による切り替えボタン



まず切り替えボタンを作成します。切り替えボタンには、a要素またはbutton要素を使用します。a要素を使用する場合は、href属性の値に切り替え対象の要素のIDやクラスを指定します（①）。また、アクセシビリティへの配慮として属性 **role="button"** を追加し、このa要素の役割がボタンであることをスクリーンリーダーなどの支援技術に伝えましょう。button要素を使用する場合は、data-target属性の値に、切り替え対象の要素のIDやクラスを指定します（②）。いずれの場合も、切り替えボタンには属性 **data-toggle="collapse"** を追加し、スクリーンリーダーなどの支援技術に対して要素の状態を伝える **aria-\*** 属性を追加します。あらかじめ表示される要素には属性 **aria-expanded="true"** を、非表示の要素には属性 **aria-expanded="false"** を追加します。また属性 **aria-controls=" (切り替える対象となる要素のIDやクラス) "** を追加し、切り替える対象であることをスクリーンリーダーなどの支援技術に伝えましょう。

次に、切り替え対象の要素に **collapse** を追加し、切り替えボタンによって表示と非表示とが切り替わるコンテンツ部分を作成します（③）。

## 7.4.2 複数の要素の表示と非表示とを切り替える

複数のa要素のhref属性、またはbutton要素のdata-target属性に、それぞれ別の値を設定することで、複数の要素の表示と非表示とを切り替えることができます。

次の例では、切り替え対象の要素のIDをdata-target属性の値に指定することで個別コンテンツの表示を切り替え、共通のクラスを指定することで複数コンテンツの表示を同時に切り替える折り畳みを作成しています（リスト7-18、図7-9）。

▼リスト7-18 複数の要素の表示と非表示とを切り替える（collapse-multiple.html）

```
<p>
    <!-- ID「content-01」の切り替えボタン -->
    <button class="btn btn-secondary" type="button" data-toggle="collapse" data-target="#content-01" ↪
        aria-expanded="false" aria-controls="content-01">ID「content-01」を表示切り替え</button>
    <!-- ID「content-02」の切り替えボタン -->
    <button class="btn btn-secondary" type="button" data-toggle="collapse" data-target="#content-02" ↪
        aria-expanded="false" aria-controls="content-02">ID「content-02」を表示切り替え</button>
    <!-- クラス「contents」の切り替えボタン -->
    <button class="btn btn-secondary" type="button" data-toggle="collapse" data-target=".contents" ↪
        aria-expanded="false" aria-controls="contents-01 contents-02">クラス「contents」を同時に表示切り替え</button>
</p>
<div class="row">
    <div class="col">
        <!-- ID「content-01」、クラス「contents」 -->
        <div class="collapse contents" id="content-01">
            <div class="card card-body">
                ID「content-01」、class「contents」
            </div>
        </div>
    </div>
</div>
```

```
<div class="col">
  <!-- ID「content-02」、クラス「contents」-->
  <div class="collapse contents" id="content-02">
    <div class="card card-body">
      ID「content-02」、クラス「contents」
    </div>
  </div>
</div>
```

▼図 7-9 複数の要素の表示と非表示とを切り替える



### 7.4.3 アコーディオンを作成する

アコーディオンとは、コンテンツのヘッダー部分をクリックすることで、コンテンツの表示と非表示を折り畳みによって切り替える機能です。Bootstrapでは、折り畳みとカード（P.124 参照）のコンポーネントを組み合わせることで、アコーディオンを作成することができます。

#### ■ 基本的なアコーディオン

次の例では、3つのカード内にヘッダーとコンテンツを配置し、折り畳みを利用してコンテンツの表示を切り替えるアコーディオンを作成しています。カードの見出しの切り替えボタンをクリックすると、コンテンツの表示・非表示が切り替えられます。また、他のカードの切り替えボタンをクリックするとコンテンツが非表示になります（リスト7-19、図7-10）。カードの作成については「カード」（P.124）を参照してください。

## ▼リスト7-19 基本的なアコーディオン (collapse-accordion.html)

```

<div class="accordion" id="accordion"> _____ ①
  <!-- カード01 -->
  <div class="card">
    <!-- カードヘッダー -->
    <div class="card-header" id="headingOne">
      <h5 class="mb-0">
        <button class="btn btn-link" type="button" data-toggle="collapse" data-target="#collapseOne" aria-expanded="true" aria-controls="collapseOne"> _____ ②
          カード01の切り替えボタン
        </button>
      </h5>
    </div>
    <!-- コンテンツ -->
    <div id="collapseOne" class="collapse show" aria-labelledby="headingOne" data-parent="#accordion"> _____ ③
      <div class="card-body">
        ...中略...
      </div>
    </div>
    </div>
  <!-- カード02 -->
  <div class="card">
    <!-- カードヘッダー -->
    <div class="card-header" id="headingTwo">
      <h5 class="mb-0">
        <button class="btn btn-link collapsed" type="button" data-toggle="collapse" data-target="#collapseTwo" aria-expanded="false" aria-controls="collapseTwo"> _____ ②
          カード02の切り替えボタン
        </button>
      </h5>
    </div>
    <!-- コンテンツ -->
    <div id="collapseTwo" class="collapse" aria-labelledby="headingTwo" data-parent="#accordion"> _____ ③
      <div class="card-body">
        ...中略...
      </div>
    </div>
    </div>
  <!-- カード03 -->
  <div class="card">
    <!-- カードヘッダー -->
    <div class="card-header" id="headingThree">
      <h5 class="mb-0">
        <button class="btn btn-link collapsed" type="button" data-toggle="collapse" data-target="#collapseThree" aria-expanded="false" aria-controls="collapseThree"> _____ ②
          カード03の切り替えボタン
        </button>
      </h5>
    </div>
  </div>

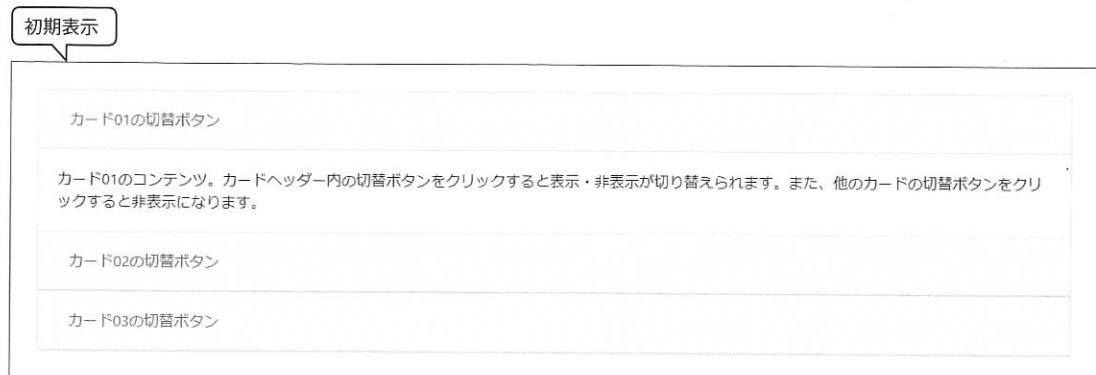
```

```

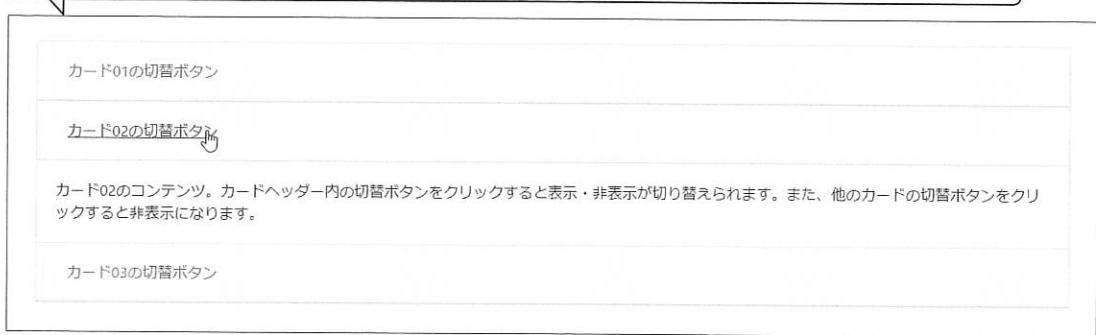
</h5>
</div>
<!-- コンテンツ -->
<div id="collapseThree" class="collapse" aria-labelledby="headingThree" data-parent="#accordion">③
  <div class="card-body">
    ...中略...
  </div>
</div>
</div>
</div>

```

▼図 7-10 基本的なアコーディオン



他のカードの切り替えボタンをクリックすると、そのカードのコンテンツが表示され、開いていたコンテンツは非表示になる



まず、アコーディオンの外枠を作成します。アコーディオンの外枠は、複数カードを含む親要素に **accordion クラス** を追加して作成します（①）。

次に、各カードヘッダーの見出し内に折り畳みの切り替えボタンを作成します。この例では、button 要素に属性 **data-toggle="collapse"** を追加します。さらに **data-target** 属性を追加し、値に切り替え対象の要素の ID を指定しています（②）。

そして、切り替えの対象となるコンテンツを作成します。コンテンツ部分は、card-body クラスが設定されたカード本文の親要素に **collapse クラス** を追加し、**data-parent** 属性の値にアコーディオンの外枠の要素の

IDを指定します（③）。このとき、初期表示されるコンテンツには**collapseクラス**が設定された要素に**showクラス**を追加します。さらに、aria-labelledby属性の値にカードヘッダーのIDを指定し、スクリーンリーダーなどの支援技術に対してカードヘッダーがこのコンテンツのラベルとして関連付けられていることを伝えましょう。

### COLUMN WAI-ARIAとは？

Bootstrapのコンポーネントで度々出てくるaria属性やrole属性は、W3Cによって定められているWAI-ARIA (<https://www.w3.org/TR/wai-aria-1.1/>) という仕様の一部です。WAI-ARIAは、「Web Accessibility Initiative - Accessible Rich Internet Applications」の頭文字をとったもので、HTMLやSVGで利用できるアクセシビリティ向上のための属性の仕様です。

この仕様では、主にrole属性とaria属性の2つが定義されています。role属性は、コンテンツの役割を定義します。例えば、role="navigation"やrole="banner"、role="search"など、要素が何なのか、何をするものなのかを定義します。

aria属性は、コンテンツの状態を表したり、性質を表すために使用します。例えば、aria-disabled="true"は、フォームのinputが現在disabledの状態であることをスクリーンリーダーに対して伝えます。また別の例で、aria-required="true"は、フォームのinputに値を入力しなければならない性質を表します。

Bootstrapの対話式コンポーネントは、タッチ、マウス、キーボード等のユーザーに対応しています。関連するWAI-ARIAのrole属性を使用することで、スクリーンリーダーなどの支援技術でも、これらのコンポーネントの役割を理解でき、操作可能になることを目指しています。

7

5

SECTION

## 折り畳みの JavaScript 使用

データ属性 API を利用せずに、JavaScript を使って折り畳みコンポーネントを有効にしたい場合は、次のように呼び出します（リスト 7-20、図 7-11）。

▼リスト 7-20 折り畳みの JavaScript 使用 (collapse-js.html)

```
<p><a class="btn btn-primary toggle-btn" href="#" role="button">a要素によるボタン</a></p>
<div class="collapse">
表示と非表示が切り替わるコンテンツ
</div>
…中略…
<script>
$('.toggle-btn').click(function(){
  $('.collapse').collapse();
});
</script>
```

▼図 7-11 折り畳みの JavaScript 使用



折り畳みコンポーネントで定義されているオプションは表 7-6 のとおりです。

▼表 7-6 折り畳みのイベント

名前	デフォルト	説明
parent	false	値には、セレクター、jQuery オブジェクト、DOM エレメントを指定。値が指定された場合、折り畳みコンテンツのうち、いずれか 1 つのコンテンツを開くと、他のコンテンツは閉じられる。 false のときは、開かれたまま閉じられない
toggle	true	呼び出し時に折り畳み可能な要素の開閉を切り替える

オプションは、データ属性または JavaScript を使用して渡すことができます。データ属性の場合、`data-parent="#sample"` のように `data-` にオプション名を追加します。

次の例では、データ属性で `parent` オプションを指定しています。`parent` オプションに値が設定されると、折り畳みコンテンツのいずれかが開かれると、他の折り畳みコンテンツは折り畳まれます（リスト 7-21、図 7-12）。

1  
2  
3  
4  
5  
6  
7  
8  
9  
10

## ▼リスト7-21 折り畳みのJavaScript使用(collapse-js-parent.html)

```

<div id="sample">
  <div class="container">
    <p>
      <a class="btn btn-secondary" data-toggle="collapse" href="#collapseContent01" role="button" aria-expanded="false" aria-controls="collapseContent01">ボタン1</a>
    </p>
    <div class="collapse" id="collapseContent01" data-parent="#sample">
      <div class="card card-body">
        コンテンツ1
      </div>
    </div>
  </div>
  <div class="container">
    ...中略...
    <div class="collapse" id="collapseContent02" data-parent="#sample">
      <div class="card card-body">
        コンテンツ2
      </div>
    </div>
  </div>
  <div class="container">
    ...中略...
    <div class="collapse" id="collapseContent03" data-parent="#sample">
      <div class="card card-body">
        コンテンツ3
      </div>
    </div>
  </div>
</div>

```

▼図7-12 データ属性でparentオプションを指定  
ボタン1、ボタン2、ボタン3の順でクリック



## 7.5.1 折り畳みのメソッド

折り畳みコンポーネントで定義されているメソッドは表7-7のとおりです。

▼表7-7 折り畳みコンポーネントのメソッド

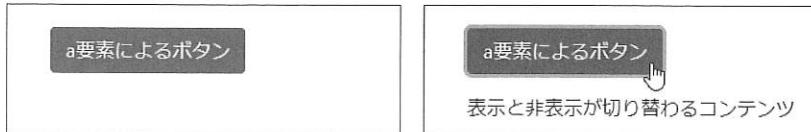
.collapse('toggle')	表示／非表示を切り替える
.collapse('show')	折り畳み可能な要素を表示する
.collapse('hide')	折り畳み可能な要素を非表示にする
.collapse('dispose')	要素の折り畳みを破棄する

次の例では、ボタンのクリックで折り畳みコンテンツの開閉を切り替えています（リスト7-22、図7-13）。

▼リスト7-22 折り畳みの表示／非表示を切り替える（collapse-method.html）

```
<script>
$('.toggle-btn').click(function(){
  $('.collapse').collapse('toggle');
});
</script>
```

▼図7-13 ボタンクリックで折り畳みコンテンツの開閉を切り替える



## 7.5.2 折り畳みのイベント

折り畳みコンポーネントでは、表7-8のようなイベントが利用できます。

▼表7-8 折り畳みのメソッド

イベントタイプ	説明
show.bs.collapse	表示される直前に発生
shown.bs.collapse	表示された直後に発生
hide.bs.collapse	非表示にされる直前に発生
hidden.bs.collapse	非表示にされた直後に発生

次の例では、hide.bs.collapseとshow.bs.collapseイベントを使用して、折り畳みコンテンツが表示・非表示にされる直前にボタンの内容を書き換えています（リスト7-23、図7-14）。

## ▼リスト7-23 折り畳みのイベント: show.bs.collapse (collapse-event.html)

```
<div class="container">
  <p><a class="btn btn-primary toggle-btn" data-toggle="collapse" href="#sample" role="button">表示する</a></p>
  <div class="collapse" id="sample">
    表示と非表示が切り替わるコンテンツ
  </div>
</div>
…中略…
<script>
$(function(){
  $('#sample').on('hide.bs.collapse', function(){
    $('.toggle-btn').html('表示する');
  });
  $('#sample').on('show.bs.collapse', function(){
    $('.toggle-btn').html('非表示にする');
  });
});
</script>
```

▼図7-14 コンテンツの表示前後でボタンキャプションを書き換え



## 6

## モーダル

ボタンをクリックするとWebページ上に表示され、ユーザーが操作を完了するまで親ウィンドウへの操作を受け付けなくさせる子ウィンドウを**モーダルウィンドウ**と言います。モーダルウィンドウは、Webページ上の画像を拡大表示するライトボックスや、ユーザー通知のためのダイアログなどに使用されます。Bootstrapの**モーダル**は、モーダルウィンドウを表示するためのコンポーネントです。

## 7.6.1 基本的な使用例 4.1

Bootstrapのモーダルは、表示の切り替えボタンとモーダルウィンドウで構成されます。

次の例では、基本的なモーダルを作成しています（リスト7-24、図7-15）。

▼リスト7-24 モーダルの基本的な使用例（modal-basic.html）

```
<!-- 切り替えボタン -->
<button type="button" class="btn btn-secondary" data-toggle="modal" data-target="#exampleModal">①
    切り替えボタン
</button>

<!-- モーダルウィンドウ外枠 -->
<div class="modal" id="exampleModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">②
    <!-- モーダルのダイアログ本体 -->
    <div class="modal-dialog" role="document">
        <!-- モーダルのコンテンツ部分 -->
        <div class="modal-content">
            <!-- モーダルのヘッダー -->
            <div class="modal-header">
                <!-- モーダルのタイトル -->
                <h5 class="modal-title" id="exampleModalLabel">モーダルのタイトル</h5>
                <!-- 閉じるアイコン -->
                <button type="button" class="close" data-dismiss="modal" aria-label="Close">③
                    <span aria-hidden="true">&times;</span>
                </button>
            </div>
            <!-- モーダルの本文 -->
            <div class="modal-body">
                モーダルの本文が入ります。
            </div>
            <!-- モーダルのフッター -->
            <div class="modal-footer">
```

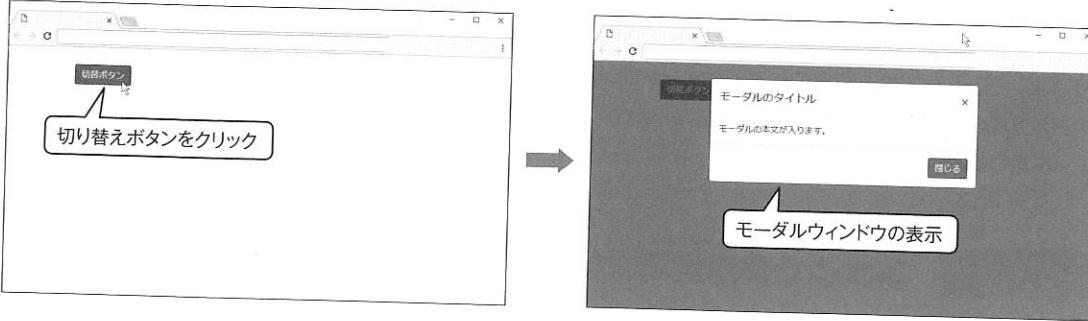
1  
2  
3  
4  
5  
6  
7  
8  
9  
10

```

<!-- 閉じるボタン -->
<button type="button" class="btn btn-secondary" data-dismiss="modal">閉じる</button> ②
</div>
</div>
</div>

```

▼図 7-15 モーダルの基本的な使用例



## 切り替えボタンの作成

モーダルウィンドウの表示を制御する切り替えボタンを作成します。**btn クラス**と**btn-{ 色の種類 } クラス**を設定した **button** 要素に、属性 **data-toggle="modal"** を追加して、JavaScript 経由でモーダルウィンドウを表示する機能を有効化します（①）。色の種類には **primary**（青）、**secondary**（グレー）などコンテクストに対応した色の種類があります。モーダルウィンドウを閉じるボタンを作成する場合は、**button** 要素に属性 **data-dismiss="modal"** を追加し、モーダルを閉じる機能を有効化します（②）。また、**data-target=" (モーダルウィンドウの ID) "** を追加して、表示切り替えの対象となる要素を指定します。

## モーダルウィンドウの作成

**div** 要素に **modal クラス**を追加し、モーダルウィンドウの外枠を作成します（③）。この要素には、切り替えの対象となる **ID** 属性と、属性 **tabindex="-1"** を設定します。これによって、キーボードの **Esc** キーでもモーダルを閉じることができます。また、アクセシビリティへの配慮として属性 **role="dialog"** を追加し、この要素の役割がダイアログであることをスクリーンリーダーなどの支援技術に伝えましょう。

モーダルウィンドウの作成に使用される定義済みクラスは表 7-9 のとおりです。

▼表 7-9 モーダルに作成するための主な定義済みクラス

クラス	概要
modal	モーダルウィンドウ外枠を形成する親要素に使用するクラス
modal-dialog	モーダルのダイアログ本体に使用するクラス
modal-content	モーダルのダイアログのコンテンツ部分に使用するクラス
modal-header	ダイアログのコンテンツ内にヘッダーを作成するためのクラス
modal-title	ダイアログのコンテンツ内にタイトルを作成するためのクラス
modal-body	ダイアログのコンテンツ本文に使用するクラス
modal-footer	ダイアログのコンテンツ内にフッターを作成するためのクラス

Bootstrap のモーダルでは、一度に 1 つのウインドウしか開くことができません。モーダルウインドウが開いている間は、親ウインドウのスクロール操作を含む機能が無効化されます。そのため、スクロールはモーダルウインドウを対象に機能します。

モーダルウインドウ外の背景をクリックすると、自動的にモーダルが閉じます。モーダルには定義済みスタイルによって position : fixed が設定され、ウインドウに対して固定配置されます。可能であれば、モーダルの HTML を入れ子にせず最上位に配置し、他の要素からの干渉を避けるようにしてください。

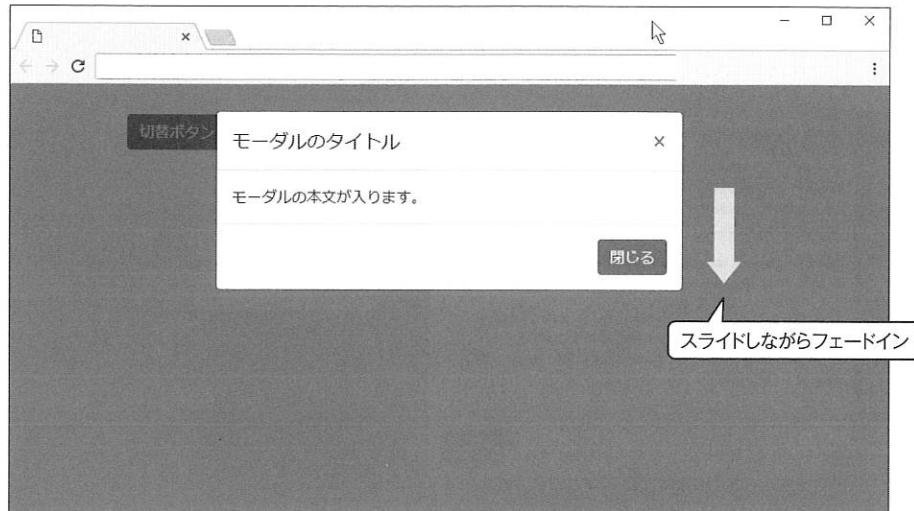
## モーダルのアニメーション設定

modal クラスが設定された要素に **fade** クラスを追加して、モーダルウインドウがページ上部からスライドしながらフェードインするアニメーションを作成することができます（リスト 7-25、図 7-16）。

▼リスト 7-25 モーダルのアニメーション設定（modal-animation.html）

```
<!-- モーダルウインドウ外枠 -->
<div class="modal fade" id="exampleModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
  ...中略...
</div>
```

▼図 7-16 モーダルのアニメーション設定



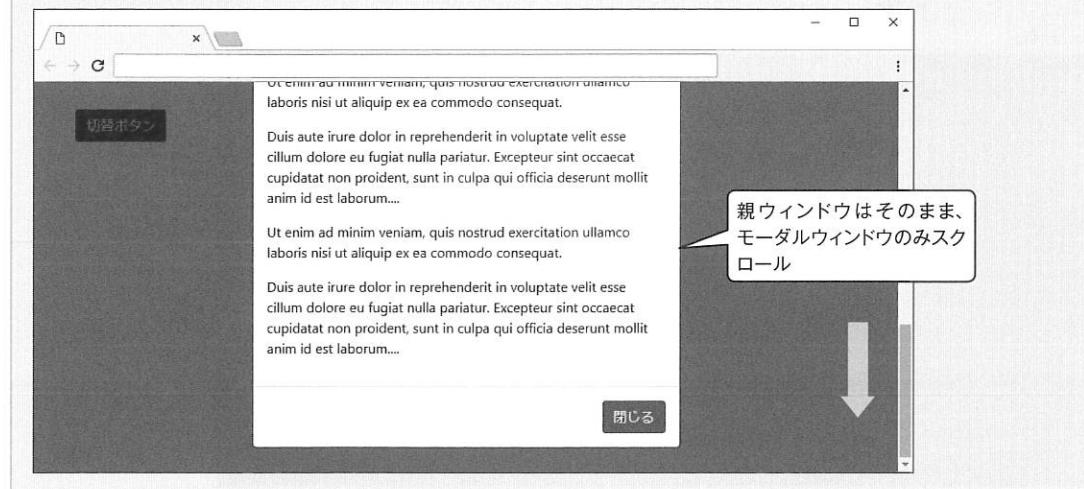
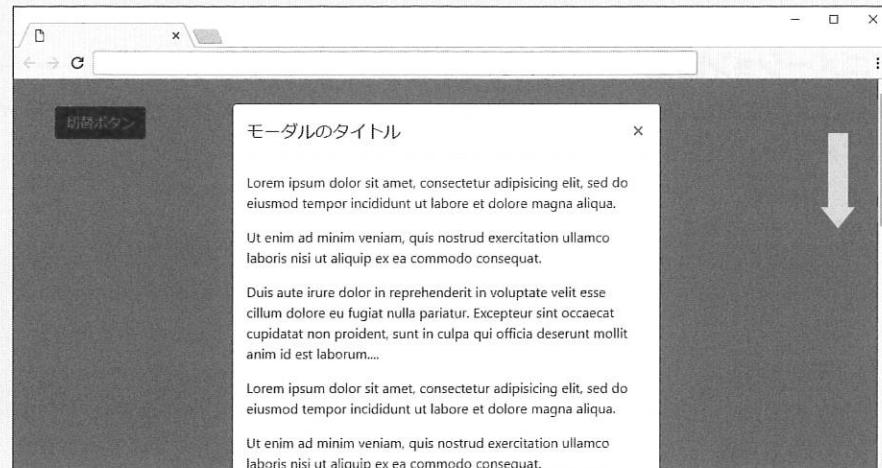
1  
2  
3  
4  
5  
6  
7  
8  
9  
10

## NOTE

## スクロールが必要な長いコンテンツのモーダル

モーダルウィンドウ内のコンテンツが長くなると、モーダルは親ウィンドウとは別に独立してスクロールします（図7-17）。

▼図7-17 長いコンテンツのモーダルの例（modal-long.html）



### ■ 垂直方向中央に配置するモーダル

モーダルは通常、親ウィンドウの上方に配置されます。この配置を変更し、モーダルをウィンドウの垂直方向中央に配置する場合、`modal-dialog` クラスを設定したダイアログ本体の要素に、**modal-dialog-centered** クラスを追加します（リスト7-26、図7-18）。

## ▼リスト 7-26 垂直方向中央に配置するモーダル (modal-dialog-centered.html)

```
<!-- モーダルウィンドウ外枠 -->
<div class="modal fade" id="exampleModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
  <!-- モーダルのダイアログ本体 -->
  <div class="modal-dialog modal-dialog-centered" role="document">
    ...中略...
  </div>
</div>
```

## ▼図 7-18 垂直方向中央に配置するモーダル



## ■ グリッドレイアウトを使用したモーダル

モーダルウィンドウ内のレイアウトに、グリッドレイアウト（P.22 参照）を使用することができます。グリッドレイアウトを使用する場合は、modal-body クラスを設定したダイアログのコンテンツ本文の要素内に、**container-fluid クラス**を設定した子要素を入れ子にし、グリッドを設定します。

次の例では、グリッドカラムの配置が見やすいように背景色の指定やスペーシングを適宜行っています（リスト 7-27、図 7-19）。

## ▼リスト 7-27 グリッドレイアウトを使用したモーダルの例 (modal-grid.html)

```
<!-- モーダルの本文 -->
<div class="modal-body">
  <div class="container-fluid">
    <!-- 1行目 -->
    <div class="row">
      <div class="col-md-4">col-md-4</div>
```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10

```
<div class="col-md-4 ml-auto">col-md-4、ml-auto</div>
</div>

<div class="row">
  <div class="col-md-3 ml-auto">col-md-3、ml-auto</div>
  <div class="col-md-2 ml-auto">col-md-2、ml-auto</div>
</div>

<div class="row">
  <div class="col-md-6 ml-auto">col-md-6、ml-auto</div>
</div>

<div class="row">
  <div class="col-sm-9">
    Level 1: col-sm-9
    <div class="row">
      <div class="col-8 col-sm-6">
        Level 2: col-8、col-sm-6
      </div>
      <div class="col-4 col-sm-6">
        Level 2: col-4、col-sm-6
      </div>
    </div>
  </div>
</div>
</div>
```

▼図 7-19 グリッドレイアウトを使用したモーダルの例



## 7.6.2 サイズのオプション 4.1

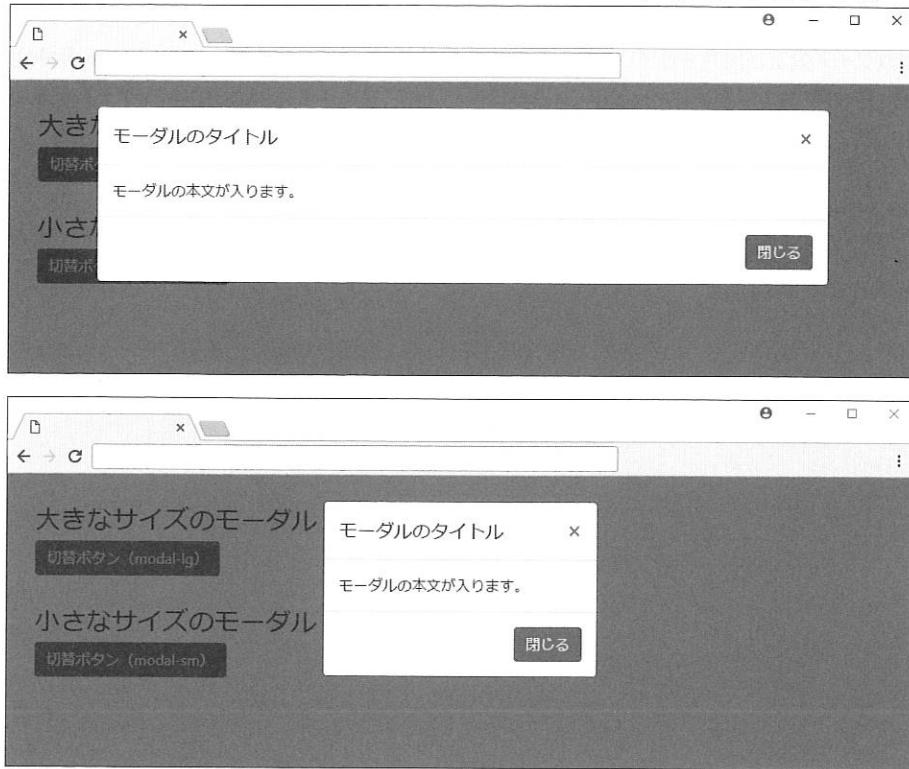
モーダルには2つのサイズ設定が用意されています。モーダルのサイズを大きくする場合は、modal-dialogクラスが設定されたダイアログ本体の要素に**modal-lg**クラスを、小さくする場合は**modal-sm**クラスを追加します（リスト7-28、図7-20）。

### ▼リスト7-28 モーダルサイズのオプション（modal-sizing.html）

```
<!-- 大きなサイズのモーダル -->
<h3>大きなサイズのモーダル</h3>
<!-- 切り替えボタン -->
<button type="button" class="btn btn-secondary" data-toggle="modal" data-target="#largeModal">
    切り替えボタン (modal-lg)
</button>
<!-- モーダル -->
<div class="modal fade" id="largeModal" tabindex="-1" role="dialog" aria-labelledby="largeModalLabel" aria-hidden="true">
    <!-- モーダルのダイアログ本体 -->
    <div class="modal-dialog modal-lg" role="document">
        ...中略...
    </div>
</div>
<!-- / 大きなサイズのモーダル -->
<!-- 小さなサイズのモーダル -->
<h3 class="mt-5">小さなサイズのモーダル</h3>
<!-- 切り替えボタン -->
<button type="button" class="btn btn-secondary" data-toggle="modal" data-target="#smallModal">
    切り替えボタン (modal-sm)
</button>
<!-- モーダル -->
<div class="modal fade" id="smallModal" tabindex="-1" role="dialog" aria-labelledby="smallModalLabel" aria-hidden="true">
    <!-- モーダルのダイアログ本体 -->
    <div class="modal-dialog modal-sm" role="document">
        ...中略...
    </div>
</div>
```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10

▼図7-20 モーダルサイズのオプション（上：大きなサイズのモーダル、下：小さなサイズのモーダル）



# モーダルの JavaScript 使用

モーダルもデータ属性 API だけでなく、JavaScript コードから使用できます。モーダルコンポーネントで定義されているオプションは表 7-10 のとおりです。オプションは、データ属性または JavaScript を使用して渡すことができます。データ属性の場合、`data-backdrop="static"` のように `data-` にオプション名を追加して指定します。

▼表 7-10 モーダルのオプション

名前	デフォルト	説明
backdrop	true	モーダルの背景をオーバーレイ表示にする。true: 有効 / false: 無効。'static' を指定するとオーバーレイ表示になるが、クリック時にモーダルを閉じない
keyboard	true	Esc キーが押されたときにモーダルを閉じる。true: 有効 / false: 無効
focus	true	初期化時にモーダルにフォーカスを移動。true: 有効 / false: 無効
show	true	初期化時にモーダルを表示。true: 有効 / false: 無効

## 7.7.1 モーダルのメソッド

モーダルで定義されているメソッドは表 7-11 のとおりです。

▼表 7-11 モーダルのメソッド

メソッド	説明
<code>.modal('toggle')</code>	表示／非表示を切り替える
<code>.modal('show')</code>	表示する
<code>.modal('hide')</code>	非表示にする
<code>.modal('handleUpdate')</code>	モーダルの高さがありスクロールバーが表示される際、モーダルの位置を手動で再調整する
<code>.modal('dispose')</code>	要素のモーダルを破棄する

次の例では、`.modal('show')` メソッドを使ってモーダルウィンドウを起動しています（リスト 7-29）。

▼リスト 7-29 `.modal('show')` メソッドを使ってモーダルウィンドウを起動（modal-method.html）

```
<script>
$('.btn').click(function(){
  $('#myModal').modal('show');
});
</script>
```

## 7.7.2 モーダルのイベント

モーダルで定義されているイベントは表7-12のとおりです。

▼表7-12 モーダルのオプション

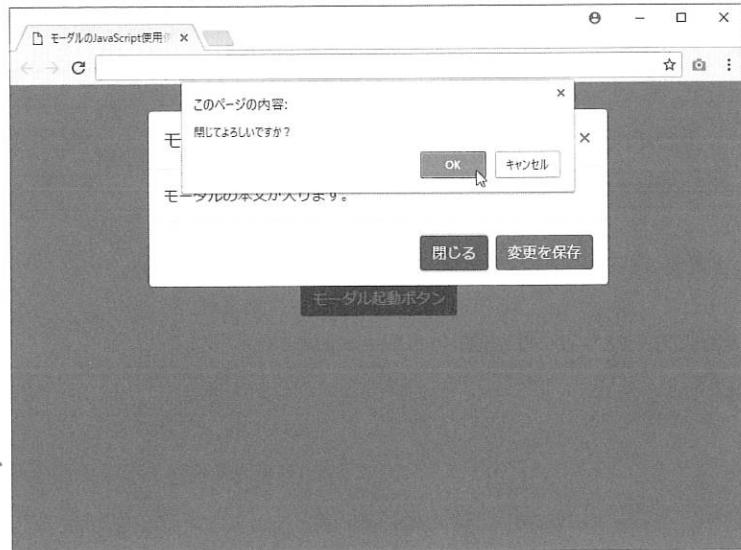
イベントタイプ	説明
show.bs.modal	表示される直前に発動。クリックによって発動した場合、そのオブジェクトがrelatedTargetプロパティに格納される
shown.bs.modal	表示された直後に発動。クリックによって発動した場合、そのオブジェクトがrelatedTargetプロパティに格納される
hide.bs.modal	非表示にされる直前に発動
hidden.bs.modal	非表示にされた直後に発動

次の例では、モーダルウィンドウが閉じられようとするときに、閉じてよいかを確認しています。[OK]を選択するとモーダルウィンドウが閉じられ、[キャンセル]が選択されると、閉じる動作をキャンセルします（リスト7-30、図7-21）。

▼リスト7-30 モーダルのイベント：hide.bs.modal (modal-event.html)

```
<script>
$('#myModal').on('hide.bs.modal', function(e){
    if(!confirm('閉じてよろしいですか？')){
        e.preventDefault(); // イベントをキャンセル
    }
});
</script>
```

▼図7-21 モーダルウィンドウが閉じられようとするときに、閉じてよいかを確認



# スクロールスパイ

Bootstrap の**スクロールスパイ**は、コンテンツのスクロール位置を監視し、該当するリンク部分を自動的にアクティブ表示させるコンポーネントです。このコンポーネントは、Bootstrap の**ナビゲーション** (P.150 参照) または**リストグループ** (P.180 参照) のコンポーネントと組み合わせて使用します。

## 7.8.1 基本的な使用例

次の例は、ナビゲーションバー内の**ナビゲーション** (P.150 参照) と組み合わせて、body 要素を監視対象としたスクロールスパイです (リスト 7-31、図 7-22)。

### ▼リスト 7-31 スクロールスパイの基本的な使用例 (scrollspy-basic.html)

```
<body data-spy="scroll" data-target="#navbar" style="position: relative; padding-top: 70px;">————①
…中略…

<!-- ナビゲーション -->
<nav id="navbar" class="navbar navbar-dark bg-dark fixed-top">
  <a class="navbar-brand" href="#">ナビゲーションバー</a>
  <ul class="nav nav-pills">
    <li class="nav-item">
      <a class="nav-link" href="#contents01">コンテンツ01</a>————
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#contents02">コンテンツ02</a>————
    </li>
    <li class="nav-item dropdown">
      <a class="nav-link dropdown-toggle" data-toggle="dropdown" href="#" role="button" ↪
         aria-haspopup="true" aria-expanded="false">コンテンツ03</a>————②
      <div class="dropdown-menu">
        <a class="dropdown-item" href="#contents03-01">01</a>————
        <a class="dropdown-item" href="#contents03-02">02</a>————
        <div role="separator" class="dropdown-divider"></div>
        <a class="dropdown-item" href="#contents03-03">03</a>————
      </div>
    </li>
  </ul>
</nav>

<!-- コンテンツ -->
<div class="mb-5">
```

```

<h4 id="contents01" style="padding-top: 60px; margin-top: -60px">コンテンツ01</h4>
…中略…
</div>

<div class="mb-5">
  <h4 id="contents02" style="padding-top: 60px; margin-top: -60px">コンテンツ02</h4>
  …中略…
</div>

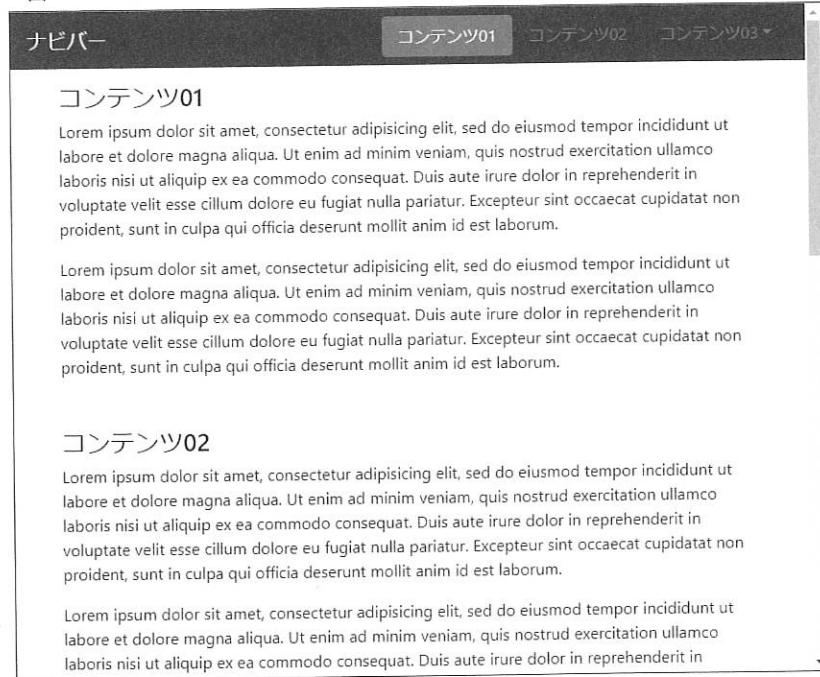
<div class="mb-5">
  <h4 id="contents03-01" style="padding-top: 60px; margin-top: -60px">コンテンツ03-01</h4>
  …中略…
</div>

<div class="mb-5">
  <h4 id="contents03-02" style="padding-top: 60px; margin-top: -60px">コンテンツ03-02</h4>
  …中略…
</div>

<div class="mb-5">
  <h4 id="contents03-03" style="padding-top: 60px; margin-top: -60px">コンテンツ03-03</h4>
  …中略…
</div>

```

▼図7-22 スクロールスパイの初期表示



まず、スクロールスパイの監視対象となるbody要素に、属性 **data-spy="scroll"** と **data-target="(ナビゲーションのID)"** を追加します（①）。監視対象となる要素には、CSSで相対配置にするためのスタイル **position:relative** が必要となります。次に、ナビゲーションのリンクをa要素で作成し、href属性の値にコンテンツの該当箇所のIDを設定します（②）。なお、この例では、上部固定のナビゲーションバーの下にコンテンツが隠れてしまわないように、body要素の上パディングに **padding-top: 70px;** を設定しています。

正常に実装されると、表示位置のコンテンツに該当するa要素には、自動的に **active** クラスが追加され、リンクがアクティブ表示になります。

スクロールした表示位置のコンテンツに該当するリンクがアクティブ表示に変わります（図7-23）。

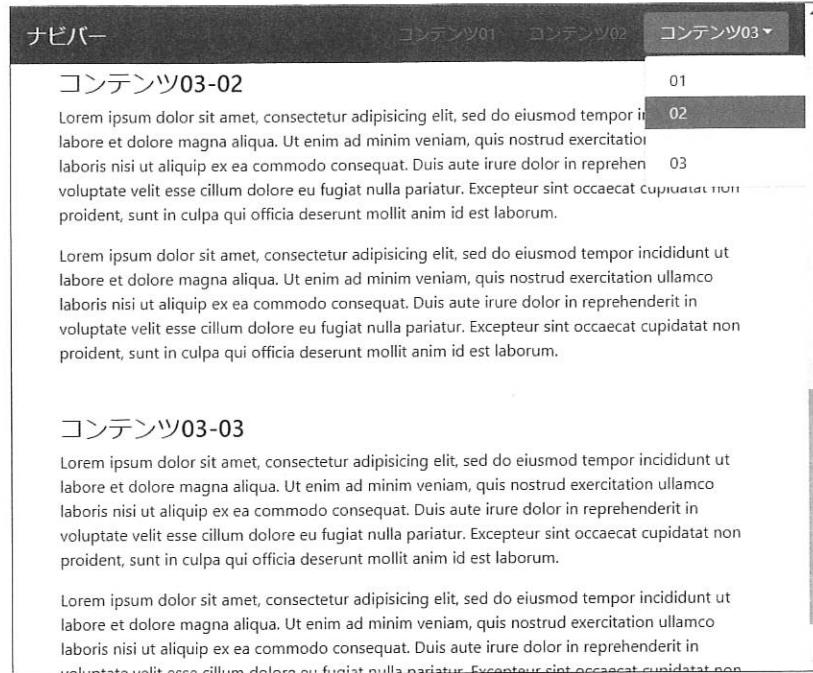
▼図7-23 表示位置に該当したリンクがアクティブ表示に変わる



入れ子になったリンクは、親リンクと合わせてアクティブ表示になります（図7-24）。

1  
2  
3  
4  
5  
6  
7  
8  
9  
10

▼図 7-24 入れ子になったリンク部分のアクティブ表示



## 7.8.2 body要素以外の要素での使用例

Bootstrap 3におけるスクロールスパイは、ナビゲーション形式でのみ使用可能でしたが、Bootstrap 4では、ナビゲーション形式に加え、**リストグループ**（P.180 参照）形式での使用も可能になりました。次の例は、グリッドレイアウトで配置した**リストグループ**と組み合わせて、特定のdiv要素を監視対象としたスクロールスパイです（リスト 7-32）。

▼リスト 7-32 body要素以外の要素にスクロールスパイを使用した例（scrollspy-listgroup.html）

```
<div class="container">
  <div class="row">
    <div class="col-4">
      <!-- リストグループ -->
      <div id="list-example" class="list-group">
        <a class="list-group-item list-group-item-action" href="#list-item-1">コンテンツ01</a>
        <a class="list-group-item list-group-item-action" href="#list-item-2">コンテンツ02</a>
        <a class="list-group-item list-group-item-action" href="#list-item-3">コンテンツ03</a>
        <a class="list-group-item list-group-item-action" href="#list-item-4">コンテンツ04</a>
      </div>
    </div>
    <div class="col-8">
```

```

<!-- コンテンツ -->
<div data-spy="scroll" data-target="#list-example" data-offset="0" class=>
"scrollspy-example border px-1"> _____
    <h4 id="list-item-1">コンテンツ01</h4> _____
    ...中略...
    <h4 id="list-item-2">コンテンツ02</h4> _____
    ...中略...
    <h4 id="list-item-3">コンテンツ03</h4> _____
    ...中略...
    <h4 id="list-item-4">コンテンツ04</h4> _____
    ...中略...
        </div>
    </div>
</div>
</div>

```

まず、スクロールスパイの監視対象となるdiv要素に、属性 **data-spy="scroll"** と **data-target="** (リ  
ストグループの ID) " を追加します (①)。body要素以外の要素を監視対象とする場合は、CSSで相対配置に  
するためのスタイル **position:relative** の設定に加え、要素の高さを決めるためのスタイル **height:\*** (この例  
では400px)、スクロールバーを表示するためのスタイル **overflow-y:scroll;** を設定する必要があります (リ  
スト7-33)。

▼リスト7-33 監視対象の要素に設定するスタイル (scrollspy-listgroup.html)

```

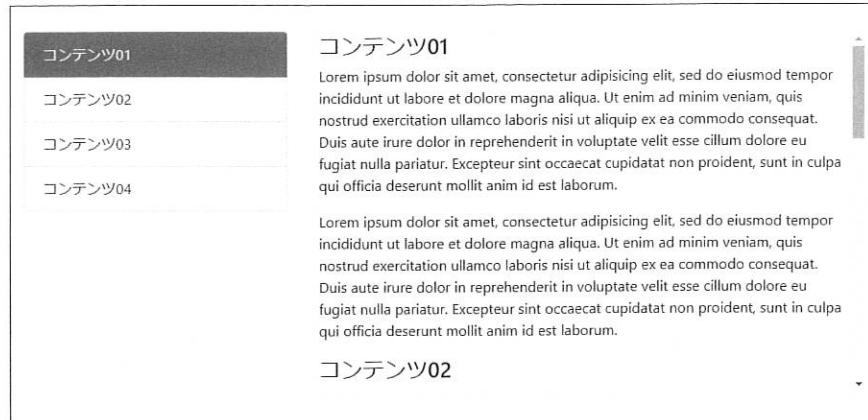
.scrollspy-example {
    position: relative;
    height: 400px;
    overflow-y: scroll;
}

```

次に、ナビゲーションのリンクをa要素で作成し、**href**属性の値にコンテンツの該当箇所のIDを設定します  
(②)。正常に実装されると、表示位置のコンテンツに該当するa要素には、自動的に**active**クラスが追加さ  
れ、リンクがアクティブ表示になります (図7-25)。

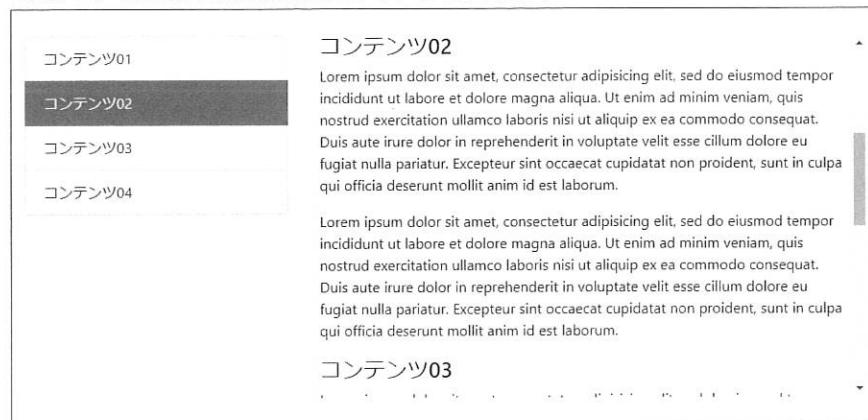
1  
2  
3  
4  
5  
6  
7  
8  
9  
10

▼図7-25 スクロールスパイの初期表示



スクロールを実行すると、スクロールした表示位置のコンテンツに該当するリンクがアクティブ表示に変わります（図7-26）。

▼図7-26 表示位置に該当したリンクがアクティブ表示に変わる



## 7

## 9

# スクロールスパイの JavaScript 使用

スクロールスパイもデータ属性 API だけでなく、JavaScript コードから使用できます。スクロールスパイで定義されているオプションは「offset」で、追従を開始するトップからのスクロール位置をピクセル単位で指定できます。デフォルトは 10 です。

オプションは、データ属性または JavaScript を使用して渡すことができます。データ属性の場合は、`data-offset="10"` のように `data-` にオプション名を追加します。

## 7.9.1 スクロールスパイのメソッド

スクロールスパイで定義されているメソッドは表 7-13 のとおりです。

▼表 7-13 スクロールスパイのメソッド

メソッド	説明
<code>.scrollspy('refresh')</code>	スクロール位置を同期し直す
<code>.scrollspy('dispose')</code>	要素のスクロールスパイを破棄

スクロールスパイを DOM の要素の追加または削除と組み合わせて使用する場合は、`refresh` メソッドを次のように呼び出す必要があります。

▼リスト 7-34 `scrollspy('refresh')` メソッド

```
+++スクリプト+++
 $('[data-spy="scroll"]').each(function () {
   var $spy = $(this).scrollspy('refresh')
 })
```

`scrollspy('refresh')` メソッドを呼び出すと、表示されている位置が再計算され、ナビゲーションバーやリストグループと同期がとられます。

## 7.9.2 スクロールスパイのイベント

スクロールスパイには、新しいアイテムがアクティブになったときに発動する `activate.bs.scrollspy` イベントが用意されています。次の例では、`activate.bs.scrollspy` イベントを使用して、アクティブなアイテムのテキストを取得し、`<h2>` に表示しています（リスト 7-35、図 7-27）。

## ▼リスト7-35 スクロールスパイのイベント:activate.bs.scrollspy (scrollspy-event.html)

```
<script>
$( '[data-spy="scroll"]' ).on('activate.bs.scrollspy', function () {
  var currentSection = $('.list-group a.active').text();
  $('h2').html(currentSection);
})
</script>
```

▼図7-27 アクティブなアイテムのテキストを取得

