HTML/CSS実習

11. Bootstrap

株式会社ジードライブ

今回学ぶ事

- CSSフレームワークのひとつであるBootstrapを使用し、画面を作成する方法

CSSフレームワークとは

- 統一的なデザインに従って作成されたCSSとJavaScriptのパッケージ
- ルールに則ってHTMLを記述するだけで見栄えの良いWebページが簡単に作成できる
  - 自身でCSSやJSを記述しなくても、レイアウトを組み、モーダルやカルーセル等を実装することができる
- CSSフレームワークの例
  - Bootstrap
  - Tailwind CSS
  - Bulma 等

Bootstrap

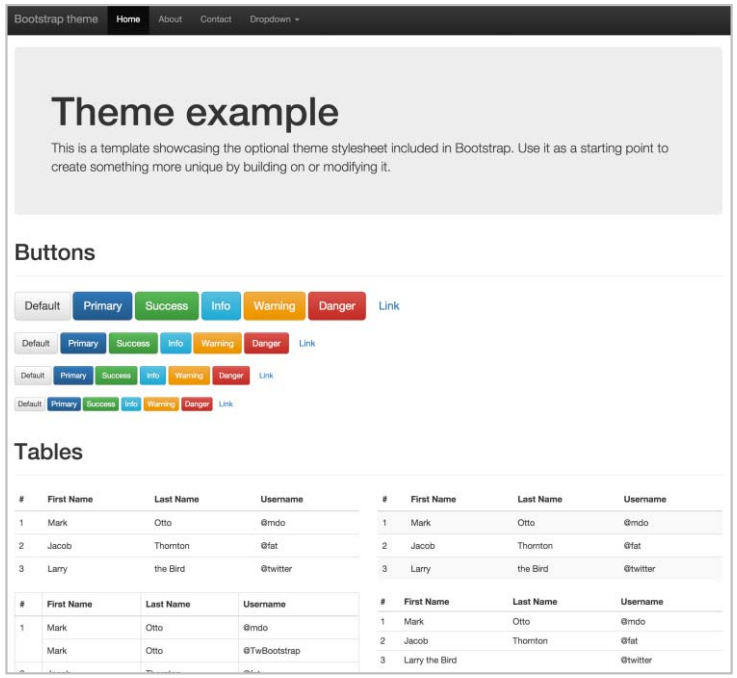
- Twitterで開発されたCSSフレームワーク
- HTMLタグに**決められたクラス名を指定する**ことで
  あらかじめ用意されたCSSが適用される
- モバイルファーストのレスポンシブサイト(スマホでの
  表示をベースにしたデザイン)を効率よく制作できる

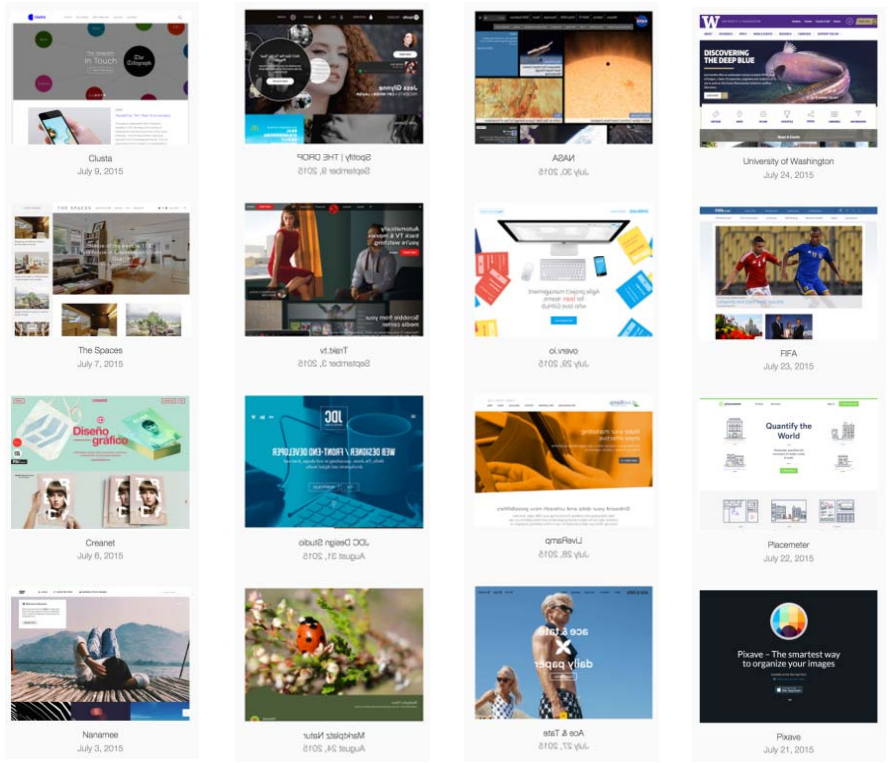
Bootstrapの利用の仕方

1. ダウンロードしたBootstrapを、そのまま利用する
   - 最低限のCSSの知識があれば、簡単に“それなりの”ページを作ることができる
   - 見栄えは整うが、オリジナリティ（サイトの独自性）はない
2. 別途CSSを追加して、カスタマイズする
   - オリジナルのデザインが可能だが、CSSの知識が必要
   - Bootstrapの設定を上書きするため、Developerツールを利用し、適切なセレクタを探り当てる技術が必要
3. Bootstrapの元ファイルをカスタマイズする
   - Sassの知識が必要
   - 難易度としては高いが、自由かつ機能的にカスタマイズできる

Bootstrapの利用の仕方

- CSSを追加したり、元ファイルをカスタマイズすることで、独自デザインに変更可能





Bootstrapを元に作られたサイトの例：https://responsive-jp.com/category/technique/bootstrap

Bootstrapの導入

Bootstrap利用の手順

1. 必要なファイルをダウンロードする
   - bootstrap.min.css, bootstrap.bundle.min.js
2. ダウンロードしたファイルを自身のHTMLに読み込む

```
<link href="bootstrap.min.css" rel="stylesheet">
<script src="bootstrap.bundle.min.js"></script>
```

3. 以下のドキュメントに従って、HTMLの階層を構成し、クラス名を設定する
   - https://getbootstrap.jp/docs/5.3/getting-started/introduction/

```
<div class="container"><div class="row">…</div></div>
<a class="btn btn-primary" href="…">ボタン</a>
```

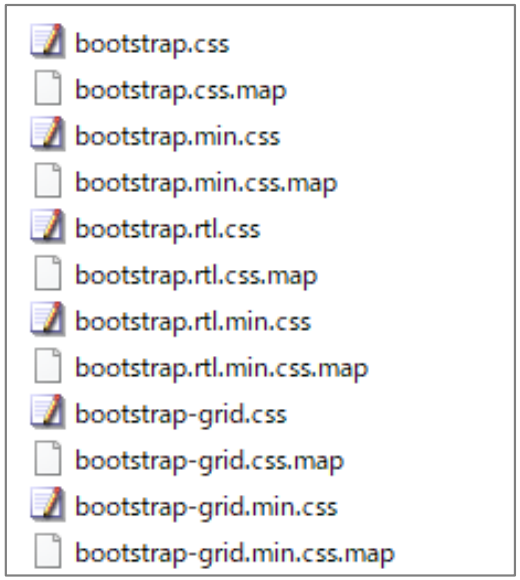
ファイルのダウンロード

- 以下からBootstrapをダウンロードする
  - https://getbootstrap.jp/docs/5.3/getting-started/download/

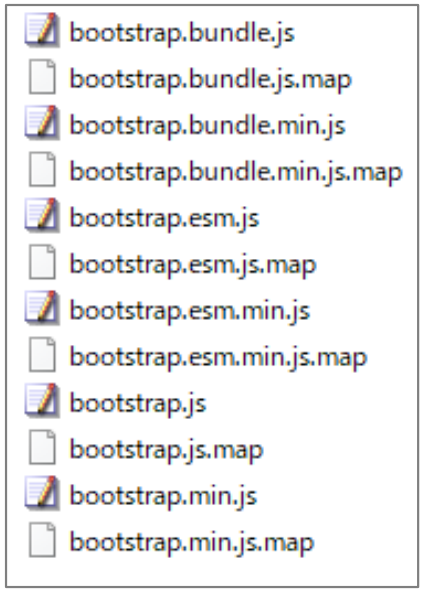
Bootstrapのファイル

- zipファイルを展開すると、cssフォルダとjsフォルダが存在し、それぞれに多くのファイルが存在することがわかる
  - ただし、これら全てを使用するわけではない

cssフォルダ内(一部)



jsフォルダ内



.mapファイルは、 CSS, JSファイルとコンパイル元のファイルとの関連を示すファイル

CSSファイル

- cssフォルダ内には以下のようなファイルが存在する

CSSファイル	説明
bootstrap-grid.css bootstrap-grid.min.css	Bootstrapのレイアウト機能を利用するためのCSSファイル
bootstrap-reboot.css Bootstrap-reboot.min.css	初期化用CSS
bootstrap.css	bootstrap-grid.cssやbootstrap-reboot.css等をひとまとめにしたもの
bootstrap.min.css	bootstrap.cssから余計な改行等を省き軽量化したもの。基本的には、これを使用する
bootstrap.rtl.css bootstrap.rtl.min.css	アラビア語など、右書きに対応するためのCSS

JavaScriptファイル

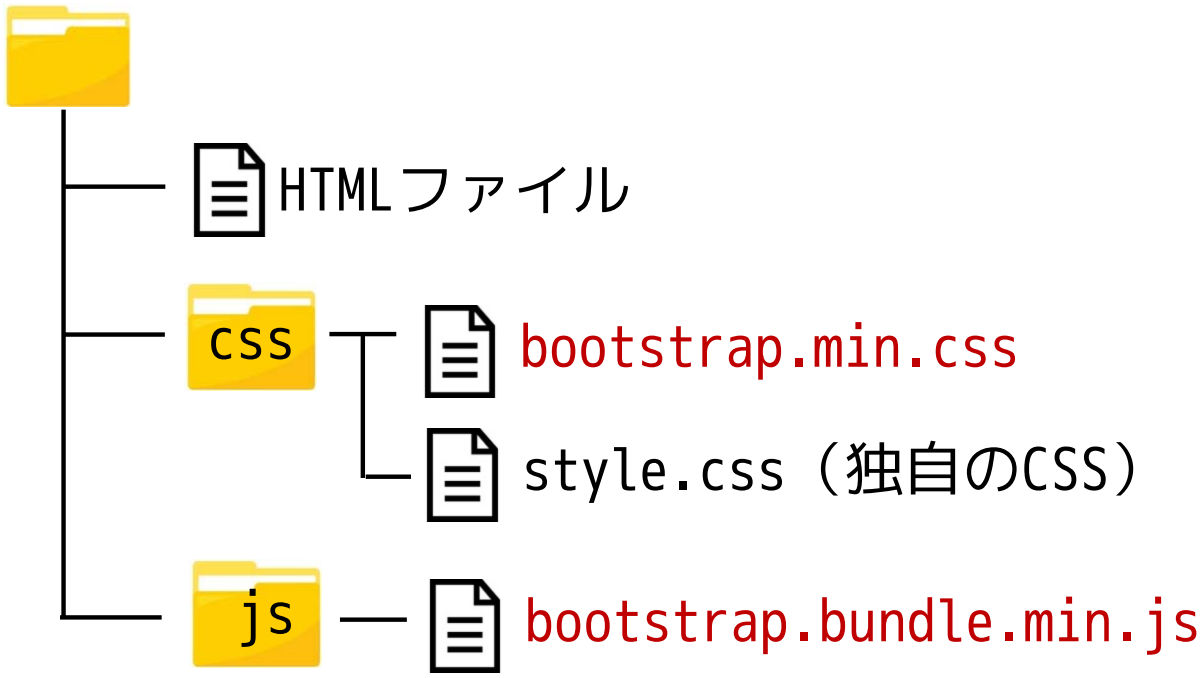
- jsフォルダ内には以下のようなファイルが存在する

CSSファイル	説明
bootstrap.js bootstrap.min.js	Bootstrapでナビゲーションなどの機能を提供するJavaScriptファイル 多くの機能でPopperというプラグインが必要 ⇒ 別途、popper.jsを読み込んで使用する
bootstrap.bundle.js	bootstrap.jsにpopper.jsを含めたもの
bootstrap.bundle.min.js	bootstrap.bundle.jsを軽量化したもの ⇒ 基本的には、これを使用する
bootstrap.esm.js bootstrap.esm.min.js	ES Moduleバージョン Bootstrapの一部の機能を、モジュールとして利用する場合に使用

Bootstrapの利用準備

- ダウンロードしたzipファイルを展開し、必要なファイルをサイト用フォルダにコピーする

ファイル／フォルダの構成例



Bootstrapの利用準備

```
<html>
<head>
<meta charset="UTF-8">
<title>ページタイトル</title>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link href="css/bootstrap.min.css" rel="stylesheet">
<link href="css/style.css" rel="stylesheet">
</head>
<body>
…
…
…
<script src="js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

スマートフォン対応に必要

Bootstrap用CSSの読み込み

独自CSSの読み込み
(BootstrapのCSSよりも後に読み込む)

Bootstrap用JavaScriptの読み込み

レイアウト

グリッドレイアウト

- グリッドレイアウトは、あらかじめ設定しておいた等間隔のガイドラインに合わせて要素を配置するデザインの手法
  - デザインの基本原則の一つである「整列」を意識的に取り入れることができる(デザインの基本原則: https://webdesign-trends.net/entry/7810)
- Bootstrapでは、グリッドレイアウトをサポートするクラスが用意されている
  - `.container / .row / .col-N` といったクラスをHTMLに付与することで、レスポンシブ対応のレイアウトを容易に行うことができる
  - 内部的にフレックスを利用したレイアウトが実装されており、水平・垂直方向の配置を柔軟に行うことができる

コンテナの設定

- コンテンツの大枠(コンテナ)は、container または container-fluid というクラス名で指定可能
  - container-fluid の場合、コンテナ幅は常に100%となる
  - container の場合、幅は以下のように変化し、中央配置される

ウィンドウのサイズ（横幅）	.containerの幅
576px未満	100%
576～767px	最大540px
768～991px	最大720px
992～1199px	最大960px
1200～1399px	最大1140px
1400px以上	最大1320px

※左右に0.75rem(初期設定では12px)ずつのパディングを含む

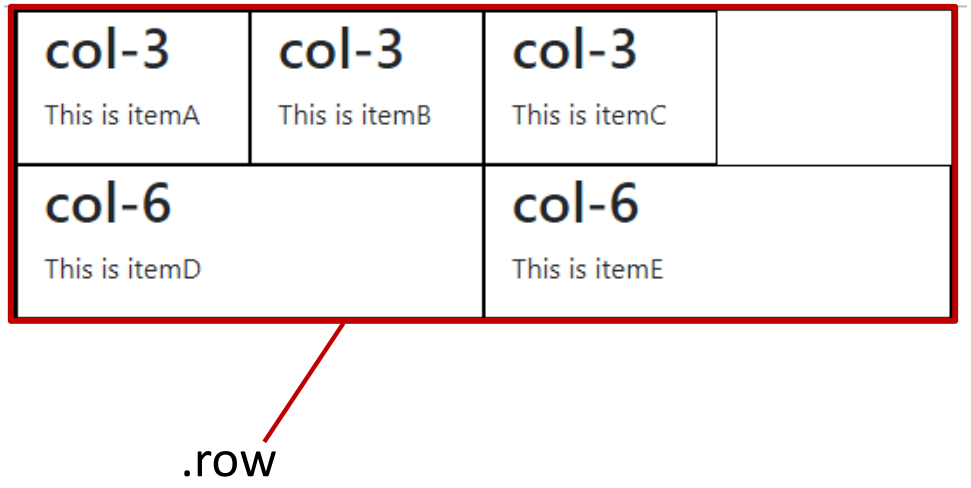
コンテナの記述例

```
<body>
<div class="container-fluid">
    <header>
        <h1>My Company</h1>
    </header>
</div>
<div class="container">
    <main>
        <section id="news">
        … 省略 …
    </main>
</div>
<script src="js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

グリッドレイアウト

- **.row**と**.col-N** (Nは1～12の整数)を組み合わせることで、グリッドに沿った横並びのレイアウトが可能
  - **.row**が親ボックスとなり、その中で**.col-N**が横に並ぶ
    (row：行)　　(column：列)
  - .rowは.container / .container-fluid内に配置する

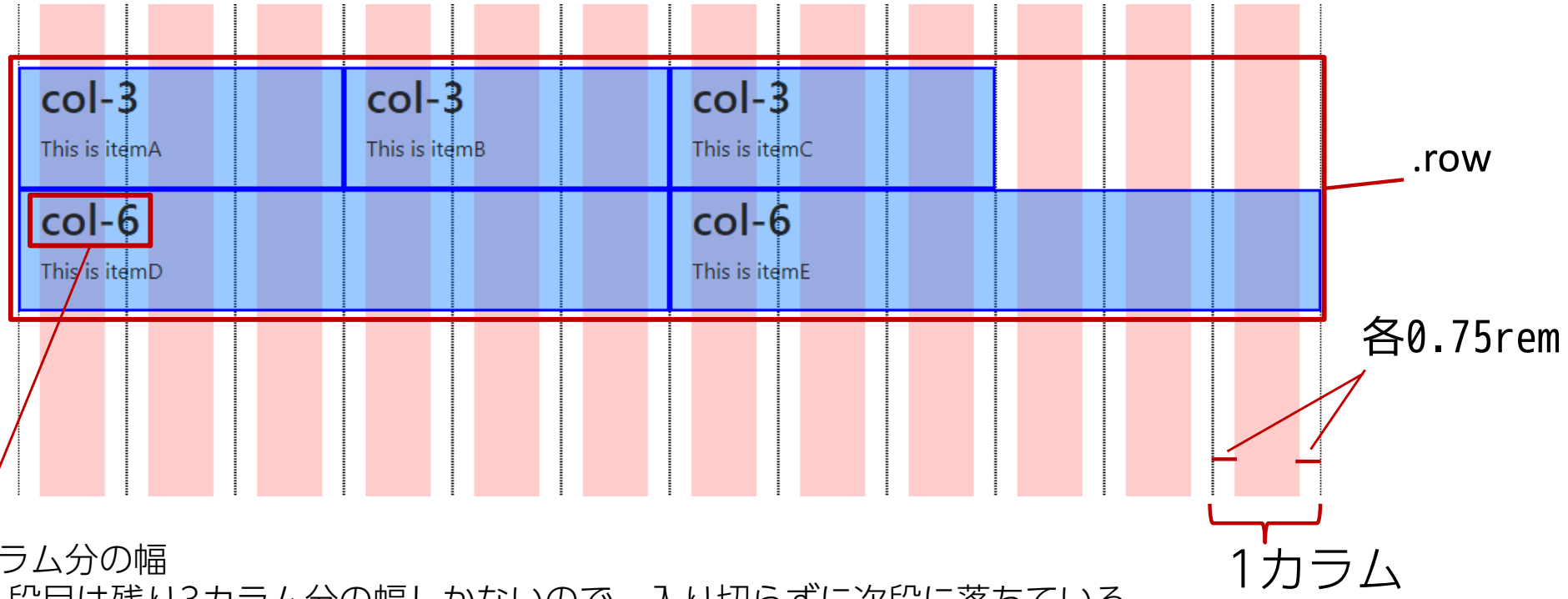
```
<div class="container">
  <div class="row">
   <div class="col-3"> itemA </div>
   <div class="col-3"> itemB </div>
   <div class="col-3"> itemC </div>
   <div class="col-6"> itemD </div>
   <div class="col-6"> itemE </div>
  </div>
</div>
```



col-の後には1～12の数字が入る(次頁参照)

グリッドとカラム

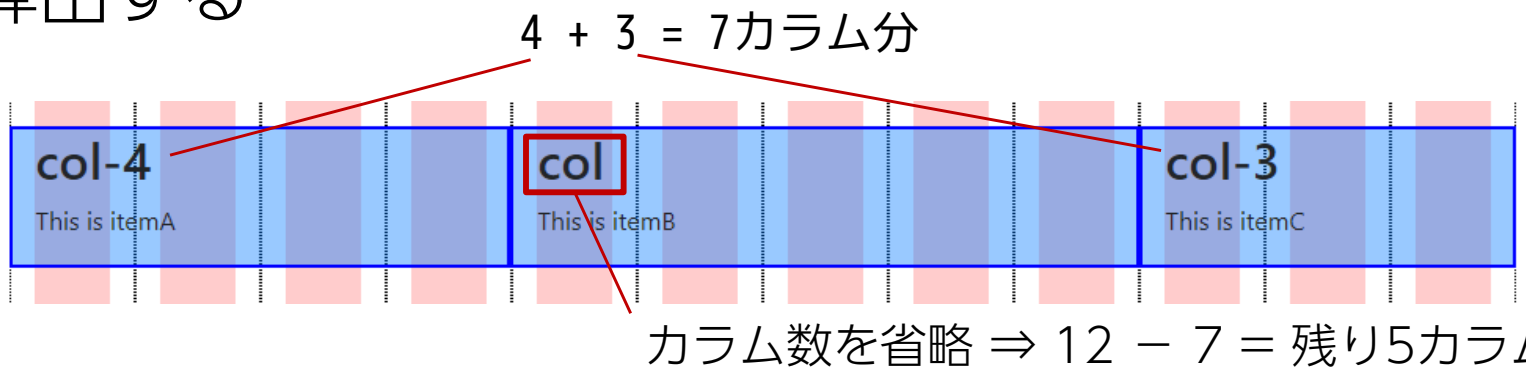
- .rowは12のカラムに分割される
  - col-N のNには、1～12のカラム数が入る(何列分の幅か指定する)
  ⇒ Nの合計が12を超える場合はカラム落ちする
  - カラムは左右に0.75rem(初期設定12px)ずつのパディングをもつ



6カラム分の幅
⇒ 1段目は残り3カラム分の幅しかないので、入り切らずに次段に落ちている

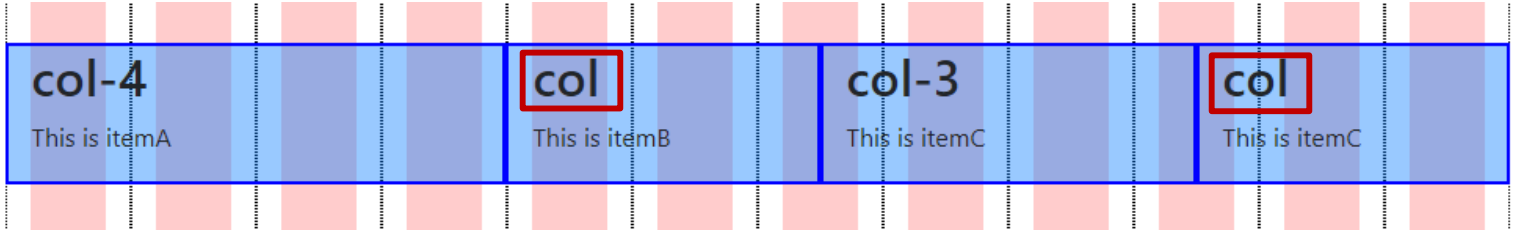
カラム数の省略

- カラム数(-N)を省略すると、空いているカラム数を自動的に算出する



– カラム数を省略したものが複数ある場合

⇒ 残りのカラム幅を均等に分ける（※最低1カラム分の幅をもつ）

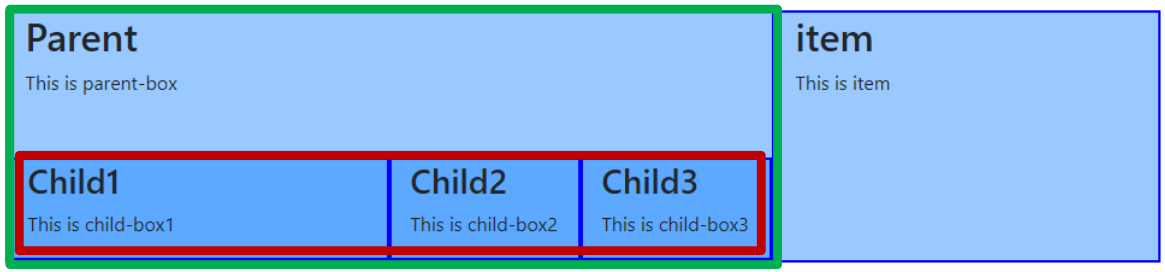


入れ子構造

- カラム内に .row を追加することで、入れ子のグリッドレイアウトを実装することができる

```
<div class="row">
      <div class="col-8">
           …Parent…
           <div class="row">
             <div class="col-6"> …Child1… </div>
             <div class="col"> …Child2… </div>
             <div class="col"> …Child3… </div>
           </div>
      </div>
      <div class="col">…item… </div>
 </div>
```

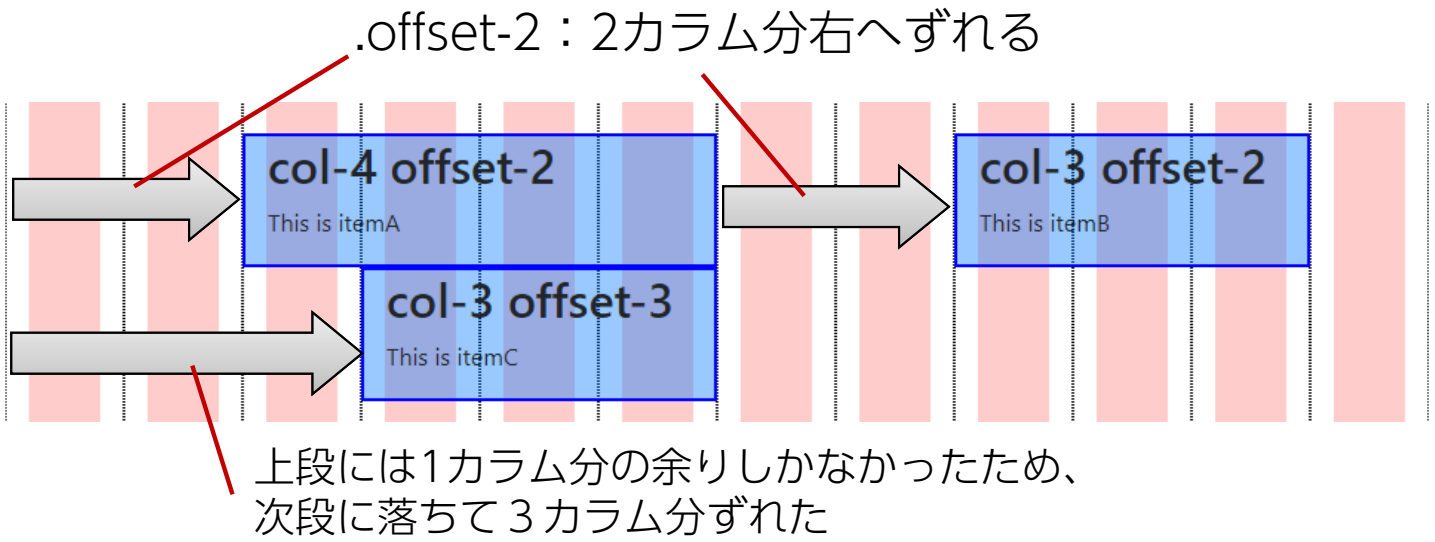
.row　カラム



オフセットの調整

- .offset-N でボックスを配置するカラムを右にずらすことができる(Nにはずらすカラム数が入る)

```
<div class="row">
        <div class="col-4 offset-2"> … itemA … </div>
        <div class="col-3 offset-2"> … itemB … </div>
        <div class="col-3 offset-3"> … itemC … </div>
</div>
```



レスポンシブ対応

- `.col-sm-3` や `.offset-md-3` のようにウィンドウ幅を示すオプションを追記することで、レスポンシブデザイン(画面幅に応じてレイアウトが変化する) に対応をすることができる

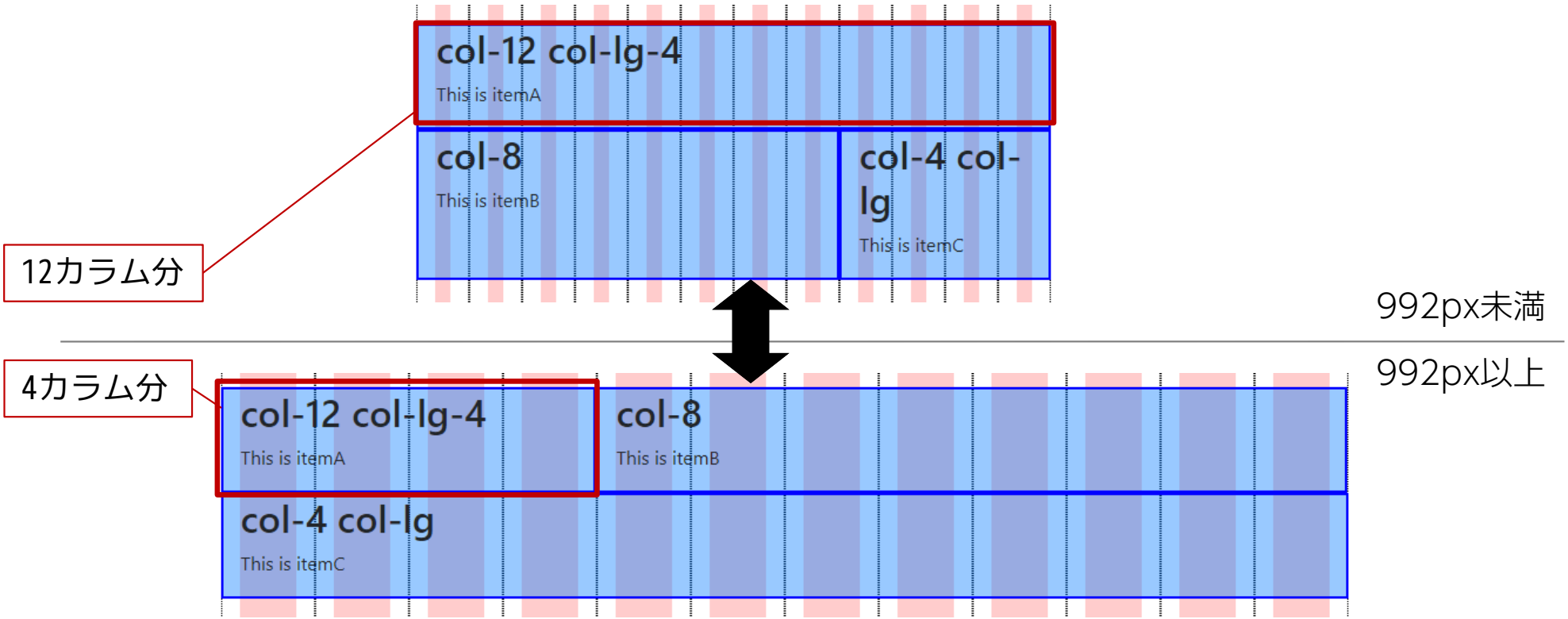
対応する幅（デバイス） ＼ クラス	なし	sm	md	lg	xl	xxl
1400px以上(特大画面)	○	○	○	○	○	○
1200px以上(大画面)	○	○	○	○	○	
992以上(PC)	○	○	○	○		
768以上(タブレット)	○	○	○			
576以上(スマホ横)	○	○				
576未満(スマホ縦)	○					

レスポンシブ対応の例

```
<div class="row">
   <div class="col-12 col-lg-4 "> … itemA … </div>
   <div class="col-8"> … itemB … </div>
   <div class="col-4  col-lg"> … itemC … </div>
</div>
```

992px以上で
4カラム分の幅になる

8カラム分の幅のまま



練習

- 練習11-1

文字の設定

文字の大きさ

- 見出しと同じ文字サイズ／太さに設定するためのクラスとして、h1～h6クラスが存在する

```
<p>株式会社ABC</p>
<p class="h1">株式会社ABC</p>
<p class="h2">株式会社ABC</p>
<p class="h3">株式会社ABC</p>
<p class="h4">株式会社ABC</p>
<p class="h5">株式会社ABC</p>
<p class="h6">株式会社ABC</p>
```

1rem(16px)
2.5rem
2rem
1.75rem
1.5rem
1.25rem
1rem

文字の大きさ

- 見出しをより目立たせたい場合、display-Nクラスを付与する（サイズが大きくなり、細字になる）
  - Nには1～6の数が入る

```
<h1>株式会社ABC</h1>
<h1 class="display-1">株式会社ABC</h1>
<h1 class="display-2">株式会社ABC</h1>
<h1 class="display-3">株式会社ABC</h1>
<h1 class="display-4">株式会社ABC</h1>
<h1 class="display-5">株式会社ABC</h1>
<h1 class="display-6">株式会社ABC</h1>
```

文字の位置(行揃え)

- 行揃えのためのクラスとしては、
  text-start(左揃え), text-center(中央揃え),
  text-end(右揃え) が用意されている

```
<div class="container text-center">
    <h1 class="display-3">株式会社ABC</h1>
    <p class="h6 text-end">代表 山田太郎</p>
    <p class="text-start">私たち株式会社ABCは、
       地域社会の発展のために創業当初から...</p>
</div>
```

文字色／背景色

- 文字色は、text-〇〇クラスで変更することができる
- 背景色は、bg-〇〇クラスで変更することができる
- 〇〇には、primary, secondar等のキーワードが入り、それぞれに割り当てられた色が反映される

```
<div class="container bg-dark">
  <h1 class="text-primary">primary</h1>
  <h1 class="text-info">info</h1>
  <h1 class="text-light">light</h1>
  <h1 class="text-warning">warning</h1>
</div>
```

色のキーワード

primary

secondary

success

danger

warning

info

white

dark

大きさ／余白の設定

大きさ

- 幅や高さを%で指定するためのクラスとして、w-〇〇 や h-〇〇 が用意されている
  - 〇〇には 25, 50, 75, 100, auto が入る
  - 幅や高さの上限を指定するためのクラスとして、mw-〇〇 や mh-〇〇 が用意されている

```
<div class="container bg-dark">
  <h1 class="w-25 bg-primary">w-25</h1>
  <h1 class="w-50 bg-success">w-50</h1>
  <h1 class="w-75 bg-warning">w-75</h1>
  <h1 class="w-100 bg-danger">w-100</h1>
</div>
```

余白

- マージンやパディングといった余白を設定するためのクラスとして、m-〇〇 や p-〇〇 クラスが用意されている
  - 〇〇には、0～5までの整数、またはautoが入る

〇〇に入る値	サイズ	ピクセル換算
0	0	0px
1	0.25rem	4px
2	0.5rem	8px
3	1rem	16px
4	1.5rem	24px
5	3rem	48px
auto	auto	-

余白

- .mt-○○やpt-○○のように、マージン及びパディングの方向を指定した書き方も可能

方向の値	意味
t	上(top)
b	下(bottom)
s	左(start)
e	右(end)

方向の値	意味
x	左右
y	上下

```
<!-- 左右の余白をautoにすることで中央に配置する -->
<div class="bg-dark">
   <div class="bg-warning w-75 mx-auto"><h1>Hello</h1></div>
</div>
```

画像の大きさ

- カラムからはみ出さないように画像を配置するには、**img-thumbnail**クラスを付与する
  - max-width: 100%; が設定される
  - 罫線と白色の余白が不要な場合は、img-fluid クラスを付与

```
<div class="col-3 bg-dark">
  <h2 class="text-light">画像の配置</h2>
  <img class="img-thumbnail  w-100"  src="…" >
</div>
```

width: 100%; が適用される
⇒ 画像よりもカラム幅が広い場合への対応

練習

- 練習11-2

テーブル

テーブル

- tableタグに、tableクラスを付与することで、各行に罫線が付くようになる

```
<table class="table">
  <thead>
        <tr><th>ID</th><th>商品名</th><th>値段</th></tr>
  </thead>
  <tbody>
        <tr><td>f001</td><td>りんご</td><td>100</td></tr>
        …
  </tbody>
</table>
```

見出し行をtheadタグで囲むことで罫線が少し濃くなる

ID	商品名	値段
f001	りんご	100
f002	みかん	300
f003	ぶどう	500

テーブル・行・セルの背景色

- tableタグに、table-〇〇クラスを付与することで、テーブルの背景色を指定できる
  - 〇〇には、primaryなどのキーワードが入る

```
<!-- テーブル全体の背景色 -->
<table class="table table-primary">
    …
</table>
```

- 色を指定するクラスは、tr, th, tdにも付与できる

```
<table class="table">
    <!- 行の背景色を指定 -->
    <tr class="table-warning">…</td>
</table>
```

交互に色を変更／マウスオーバー

- tableタグに、table-stripedクラスを付与することで、各行の背景色の交互に変更される(濃淡が付く)

```
<table class="table table-striped">…</table>
```

ID	商品名	値段
f001	りんご	100
f002	みかん	300
f003	ぶどう	500

- 行をマウスオーバーした際に、その行をハイライトするには、table-hoverクラスを付与する

```
<table class="table table-hover">…</table>
```

フォーム

テキスト入力欄

- form-controlクラスを付与することで、テキスト入力欄をカラム幅に合わせて広げることができる
  - form-control-lg や form-control-sm も併せて付与することで、入力欄の高さを調整できる
- labelタグにはform-labelクラスを付与する

```
<div class="row">
  <div class="col-12 mb-3">
    <label class="form-label" for="userName">ユーザー名</label>
    <input class="form-control form-control-lg" type="text" id="userName">
  </div>
  <div class="col-12">
    <label class="form-label" for="message">メッセージ</label>
    <textarea class="form-control" rows="3" id="message"></textarea>
  </div>
</div>
```

ラベルを横に配置する際の位置調整

- labelタグにcol-form-labelタグを付与することで、ラベルと入力欄を横に並べた際の位置調整が行われる

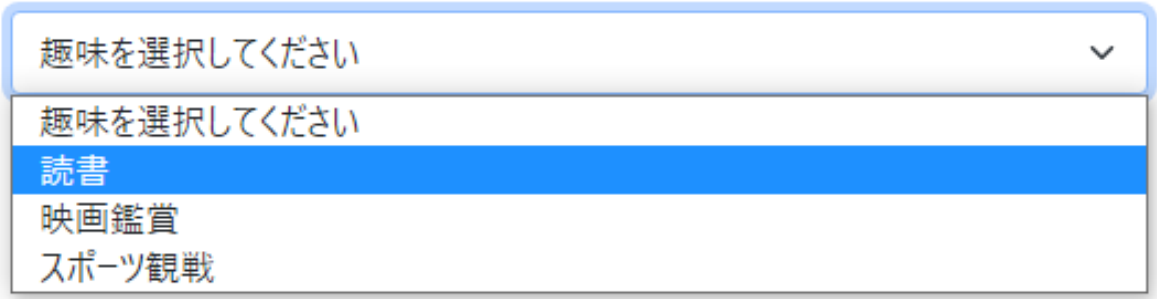
```
<!-- ラベルと入力欄を横に並べる -->
<div class="row">
  <label class="col-2 col-form-label" for="userName">ユーザー名</label>
  <div class="col-10 mb-3">
    <input class="form-control" type="text" id="userName">
  </div>
  <label class="col-2 col-form-label" for="message">メッセージ</label>
  <div class="col-10">
    <textarea class="form-control" rows="3" id="message"></textarea>
  </div>
</div>
```

ラベルと入力欄がきれいに並ぶ

select要素

- select要素には、form-selectクラスを付与する

```
<select class="form-select">
    <option selected>趣味を選択してください</option>
    <option value="1">読書</option>
    <option value="2">映画鑑賞</option>
    <option value="3">スポーツ観戦</option>
</select>
```



チェックボックス/ラジオボタン

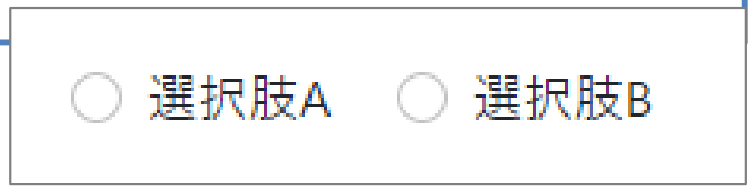
- チェックボックス/ラジオボタンには、form-check-inputクラスを付与する
  - ラベルには、form-check-labelクラスを付与する
  - 全体をform-checkクラスを付与したdivタグで囲む

```
<div class="form-check">
    <input class="form-check-input" type="checkbox" id="choiceA">
    <label class="form-check-label" for="choiceA">選択肢A</label>
</div>
<div class="form-check">
    <input class="form-check-input" type="checkbox" id="choiceB">
    <label class="form-check-label" for="choiceB">選択肢B</label>
</div>
```

チェックボックス/ラジオボタン

- 要素を横に並べる場合は、form-check-inlineを付与する

```
<div class="form-check form-check-inline">
  <input class="form-check-input" type="radio" value="A" id="choiceA">
  <label class="form-check-label" for="choiceA">選択肢A</label>
</div>
<div class="form-check form-check-inline">
  <input class="form-check-input" type="radio" value="B" id="choiceB">
  <label class="form-check-label" for="choiceB">選択肢B</label>
</div>
```



練習

- 練習11-3

コンポーネント

コンポーネント

- ページ内に配置する**部品**のことをコンポーネントと呼ぶ
  - ボタン
  - ページネーション（ページ番号）
  - ナビゲーションバー
  - アラート
  - モーダル（ボタンクリック等で呼び出される表示領域）
  - カルーセル（循環する画像のスライド）…等々

ボタン

- .btnと.btn-〇〇 の２つのクラスでボタンのスタイルを形成する
  - 〇〇には primary等が入り、色を指定することができる
  - フォームのボタン、button要素、a要素に適用可

```
<input type="submit" class="btn btn-primary" value="Click">
```

```
<button class="btn btn-primary">Click</button>
```

```
<a href="#" class="btn btn-primary">Click</a>
```

Click

ボタンの色

Primary Secondary Success Danger Warning Info Light Dark

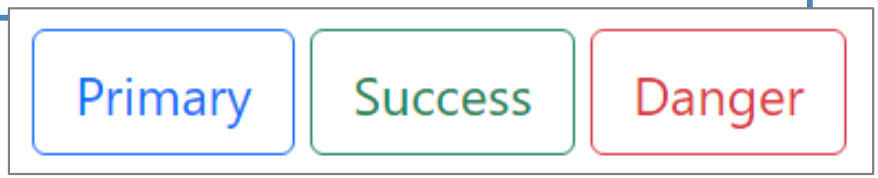
ボタン

- ボタンのサイズは、btn-sm や btn-lgクラスで変更できる

```
<a href="#" class="btn btn-primary btn-sm">Small</a>
<a href="#" class="btn btn-primary">Default</a>
<a href="#" class="btn btn-primary btn-lg">Large</a>
```

- クラス名にoutlineを入れると、枠線のみのデザインになる

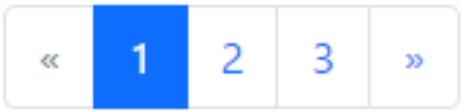
```
<a href="#" class="btn btn-outline-primary">Primary</a>
<a href="#" class="btn btn-outline-success">Success</a>
<a href="#" class="btn btn-outline-danger">Danger</a>
```



ページネーション

- ページネーション(ページ番号)を実装するには、以下のようなクラスを付与する
  - ulタグ：paginationクラス
  - liタグ：page-itemクラス
  - aタグ：page-linkクラス
  - リンクを無効にする：disabledクラス
  - 現在のページ：activeクラス

```
<ul class="pagination">
  <li class="page-item disabled">
      <a class="page-link" href="#">&laquo;</a></li>
  <li class="page-item active"><a class="page-link" href="#">1</a></li>
  <li class="page-item"><a class="page-link" href="#">2</a></li>
  <li class="page-item"><a class="page-link" href="#">3</a></li>
  <li class="page-item"><a class="page-link" href="#">&raquo;</a></li>
</ul>
```



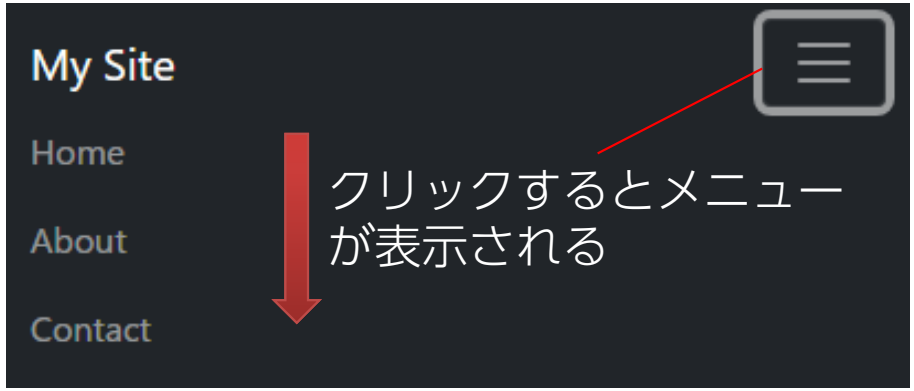
練習

- 練習11-4

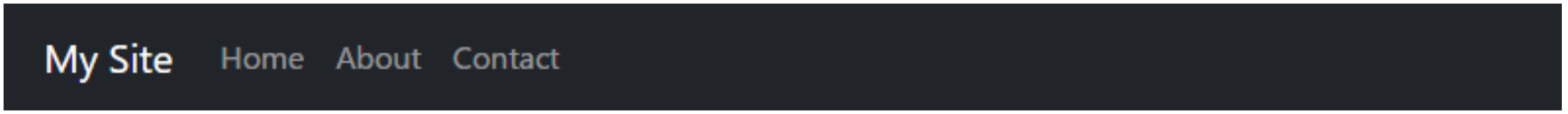
ナビゲーションバー

- 画面幅に応じて、バーガーアイコンに折り畳み可能なナビゲーション

画面幅が狭い場合、バーガーアイコンが表示される



ウィンドウを広げる



画面幅が広がるとバーガーアイコンが消え、メニューが表示される

ナビゲーションバーの作成①

- navタグに以下のクラスを付与する
  - navbar：ナビゲーションバーであることを示す
  - navbar-expand-〇〇：〇〇にはmd, lgといったサイズが入る
    ⇒ どのサイズから折り畳みを解除するかを指定
  - navbar-〇〇：文字色の指定。〇〇には light または darkが入る
    ⇒ 背景色が明るい場合はlight、背景色が暗い場合にはdark
  - .bg-〇〇：背景色の指定。〇〇には primary や dark 等が入る

```
<nav class="navbar navbar-expand-md navbar-dark bg-dark">
    <div class="container">
        次頁で追記
    </div><!-- /.container -->
</nav>
```

または container-fluid

ナビゲーションバーの作成②

- navbar-brandクラス：サイト名を囲むaタグに付与
- navbar-toggler-iconクラス： buttonタグに付与
  ⇒ data-bs-toggle属性として、collapseを設定
  ⇒ data-bs-target属性は、折り畳む要素のidと一致させる
- navbar-toggler-iconクラス：spanタグに付与
  ⇒ バーガーアイコンを背景画像として読み込む

```
<div class="container">
  <a class="navbar-brand" href="#">My Site</a>
  <button class="navbar-toggler"
          data-bs-toggle="collapse" data-bs-target="#menu">
       <span class="navbar-toggler-icon"></span>
  </button>
  次頁で追記
</div><!-- /.container -->
```

ナビゲーションバーの作成③

- collapse/navbar-collapseクラス：折り畳むdiv要素に付与
  ⇒ id属性は、buttonタグのdata-bs-target属性と一致させる
- ul：navbar-navクラス, li：nav-itemクラス, a：nav-linkクラスをそれぞれ付与する

```
    ...
    </button>
    <div class="collapse navbar-collapse" id="menu">
      <ul class="navbar-nav">
        <li class="nav-item">
          <a class="nav-link active" href="#">Home</a></li>
        <li class="nav-item">
          <a class="nav-link" href="#">About</a></li>
      </ul>
    </div><!-- /.collapse.navbar-collapse -->
</div><!-- /.container -->
```

固定ナビゲーションバー

- 上部に固定

```
<nav class="navbar …… fixed-top">
```

- 下部に固定

```
<nav class="navbar …… fixed-bottom">
```

- スクロールで最上部に来た時に固定

```
<nav class="navbar …… sticky-top">
```

アラート

- アラートはエラーメッセージなどを表示させる際に利用するコンポーネント

primaryやwarningなども設定可

```
<div class="alert alert-danger">
  パスワードを入力してください
</div>
```

- 閉じるためのボタンを付けることもできる

```
<div class="alert alert-danger alert-dismissible fade show">
  パスワードを入力してください
  <button class="btn-close" data-bs-dismiss="alert"></button>
</div>
```

練習

- 練習11-5