Bootstrap

Paweł Staniszewski | logintegra

Wyższa Szkoła Bankowa w Gdańsku

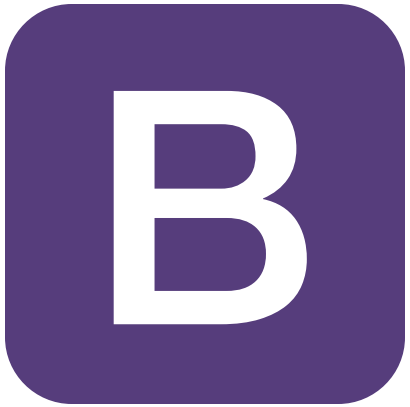
Plan spotkania

1. Czym jest *Bootstrap* i dlaczego go używamy.
2. Instalacja *Bootstrapa*.
3. *Media queries* (bardzo krótko).
4. *Grid system*.
5. Marginesy.
6. Ćwiczenie!
7. Tabele, listy, klasy pomocnicze.
8. **Formularze** – duża sekcja.
9. Czysty `CSS` – `CSS Grid`, `Flexbox` (jeśli starczy nam czasu).
10. Ćwiczenie / praca domowa – formularz z walidacją.

Czym jest *Bootstrap*?

> Quickly design and customize responsive mobile-first sites with Bootstrap, the world's most popular front-end open source toolkit, featuring Sass variables and mixins, responsive grid system, extensive prebuilt components, and powerful JavaScript plugins.

https://getbootstrap.com/



Czym jest *Bootstrap* wg Wikipedii?

> Zawiera zestaw przydatnych narzędzi ułatwiających tworzenie interfejsu graficznego stron oraz aplikacji internetowych. Bazuje głównie na gotowych rozwiązaniach HTML oraz CSS (kompilowanych z plików Less[2]) i może być stosowany m.in. do stylizacji takich elementów jak teksty, formularze, przyciski, wykresy, nawigacje i innych komponentów wyświetlanych na stronie. Biblioteka korzysta także z języka JavaScript.

https://pl.wikipedia.org/wiki/Bootstrap_(framework)

Bootstrap – Instalacja

Najprostszym sposobem jest dodanie do dokumentu `html` kilku importów:

- `bootstrap.min.css` – plik `css`:

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta1/dist/css/bootstrap.min.css" rel="stylesheet" />
```

- `bootstrap.bundle.min.js` – `javascript` wraz z biblioteką `Popper` do obsługi

```
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta1/dist/js/bootstrap.bundle.min.js"></script>
```

Przykładowy dokument można skopiować ze strony Getting started:

Starter template

Be sure to have your pages set up with the latest design and development standards. That means using an HTML5 doctype and including a viewport meta tag for proper responsive behaviors. Put it all together and your pages should look like this:

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta1/dist/css/bootstrap.min.css" rel
```

Copy

Kod wklejamy do pustego pliku, zapisujemy np. jako `index.html` i otwieramy w przeglądarce.

- `<meta charset="utf-8">`
  - kodowanie strony – `UTF-8` jest standardem i zawsze dobrym wyborem;
- `<html lang="en">` :
  - określa główny język strony;
  - pomaga czytnikom (*screen readers*);
- `<meta name="viewport" content="width=device-width, initial-scale=1">` :
  - https://www.w3schools.com/css/css_rwd_viewport.asp;
  - niezbędny do poprawnego działania na urządzeniach mobilnych;
  - pozwala dostosować stronę do rozmiaru urządzenia mobilnego.

Przygotowanie przy pomocy `css` strony zawierającej trzy kolumny o różnych szerokościach jest zaskakująco skomplikowane
— w3schools.com.

```
* {
  box-sizing: border-box;
}
.column {
  float: left;
  padding: 10px;
}
.left,
.right {
  width: 25%;
}
.middle {
  width: 50%;
}
```

Przykład na Moodle – `three-columns-css.html`.

Trzy kolumny różnej szerokości przy użyciu *Bootstrap*

Analogiczny efekt trzech kolumn możemy uzyskać dzięki *Bootstrap* dzięki kilku klasom:

```
<div class="container">
  <div class="row">
    <div class="col-2">Kolumna 1</div>
    <div class="col-8">Kolumna 2</div>
    <div class="col-2">Kolumna 3</div>
  </div>
</div>
```

Breakpointy* - punkty (szerokości) zmiany układu strony

Breakpoint	Infiks (wrostek) klasy	Wymiary
X-Small	-	<576px
Small	`sm`	≥576px
Medium	`md`	≥768px
Large	`lg`	≥992px
Extra large	`xl`	≥1200px
Extra extra large	`xxl`	≥1400px

* Wybaczcie to okropne słowo, ale nie znam dobrego polskiego odpowiednika.

Media queries

Bootstrap przy ustalaniu breakpointów korzysta
z media queries. Pozwalają one
określić zakres działania kodu `css` w zależności od np. szerokości ekranu.

`min-width` oraz `max-width`

Poniższy kod pozwoli zastosować styl `display: none` (ukrycie) do elementów z klasą `my-class` tylko w przypadku strony
o szerokości między 768px a 1200px.

```
@media (min-width: 768px) and (max-width: 1200px) {
  .my-class {
    display: none;
  }
}
```

Kontenery

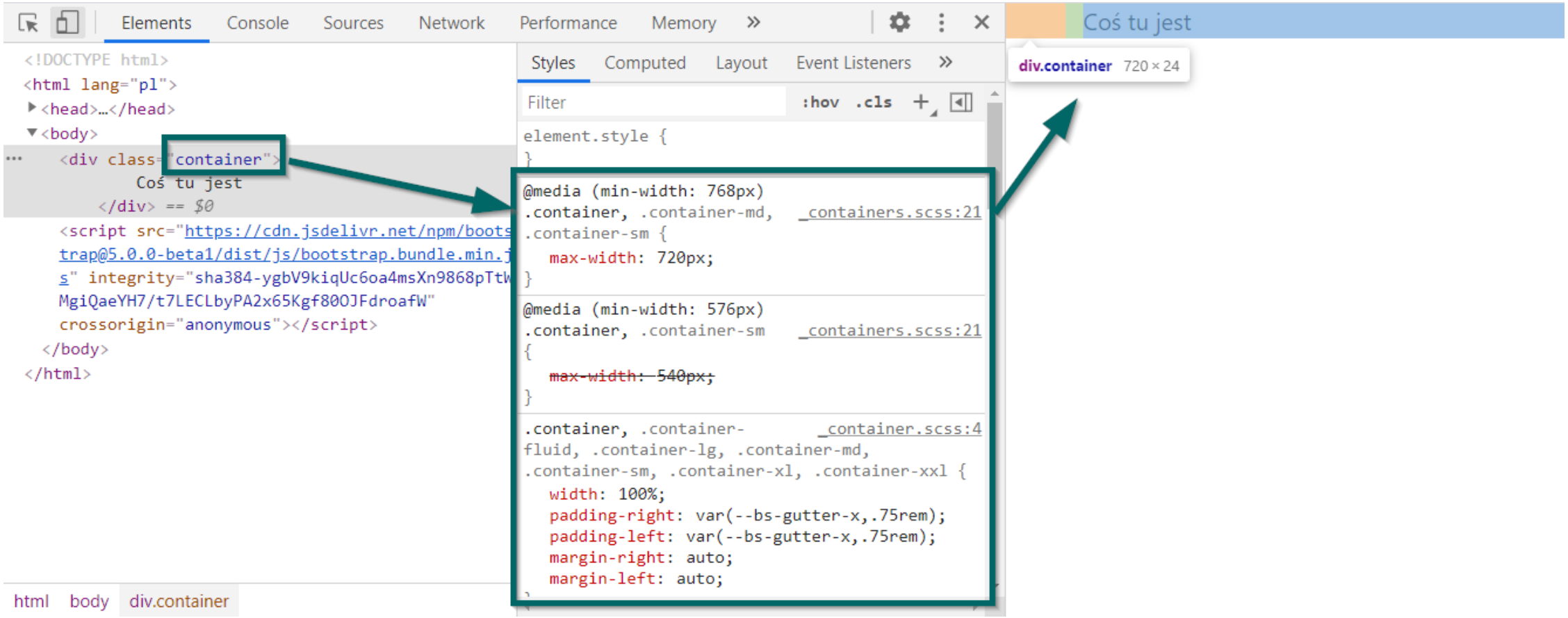
Kontenery – containers – są niezbędne do działania siatki (*grid layout*).

- `container` – maksymalna szerokość na każdym breakpoincie;
- `container-fluid` – zawsze zajmuje pełną szerokość – `width: 100%`;
- `container-{breakpoint}` – pełna szerokość do czasu osiągnięcia danego breakpointu.

Przykład

```
<div class="container">
  <!-- Pozostały kod -->
</div>
```

Kontenery – podstawowy przykład



Grid system

Być może najbardziej wartościowym elementem Bootstrap jest jego grid system – pozwalający budować złożone, responsywne, działające na urządzeniach mobilnych siatki elementów.

Przykład – trzy kolumny równej szerokości

```
<div class="container">
  <div class="row">
    <div class="col">col</div>
    <div class="col">col</div>
    <div class="col">col</div>
  </div>
</div>
```

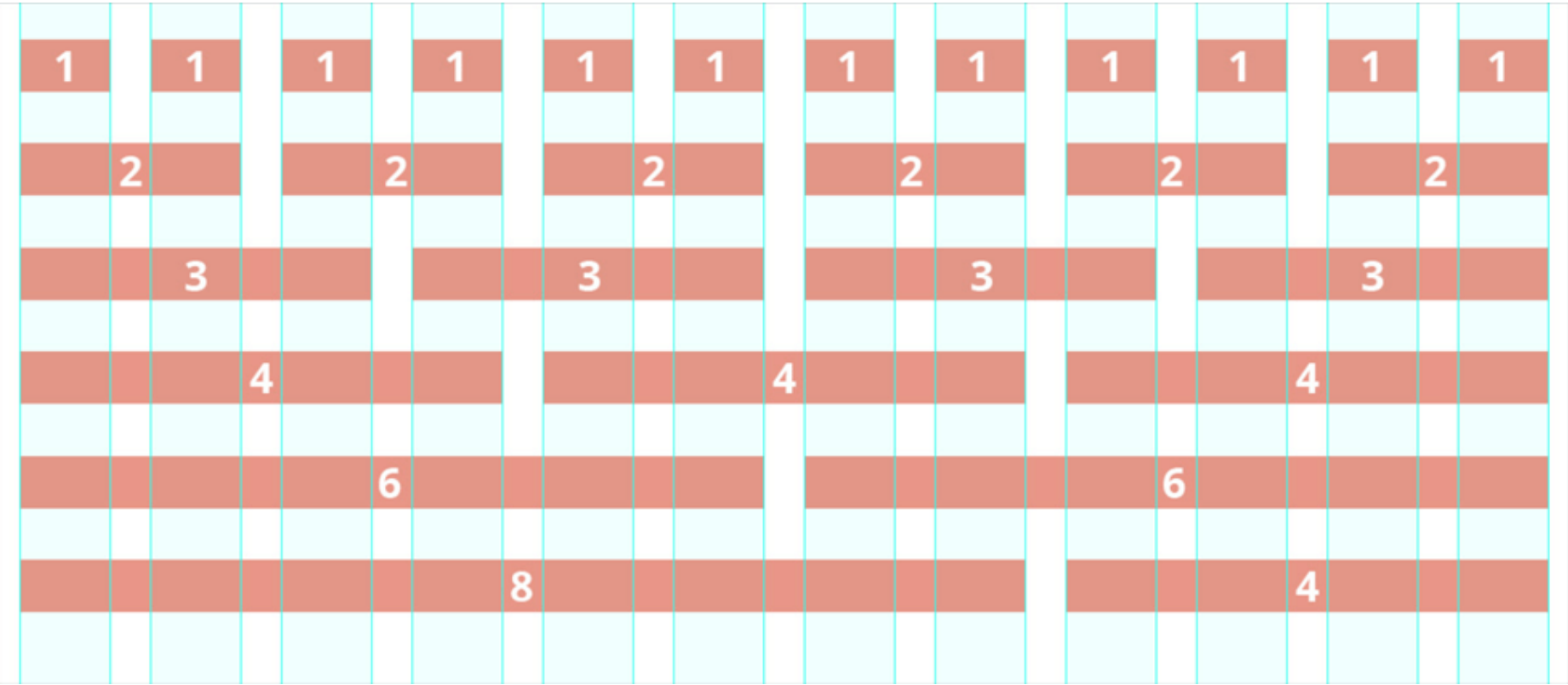
Przykład z dokumentacji – zbuduje trzy kolumny równej szerokości na wszystkich urządzeniach.

Kolumny o równej szerokości – `.col`

Jeżeli chcemy mieć wybraną liczbę kolumn o równej szerokości, korzystamy z klasy `col`:

```
<div class="container">
  <div class="row">
    <div class="col">1/5</div>
    <div class="col">1/5</div>
    <div class="col">1/5</div>
    <div class="col">1/5</div>
    <div class="col">1/5</div>
  </div>
</div>
```

Bootstrap dzieli każdy wiersz (`.container > .row`) na **12 bazowych kolumn**:



Obrazek: https://codelikethis.com/lessons/bootstrap/bootstrap-development

Ustalamy szerokość niektórych kolumn

Możemy sprawić, by niektóre z kolumn miały określoną szerokość – np. zajmowały 1/4 dostępnej szerokości – a pozostałe
się dopasowały:

```
<div class="container">
  <div class="row">
    <div class="col">Szerokość automatyczna</div>
    <div class="col-4">1/3 szerokości kontenera</div>
    <div class="col">Szerokość automatyczna</div>
    <div class="col-4">1/3 szerokości kontenera</div>
  </div>
</div>
```

Szerokość automatyczna	1/3 szerokości kontenera	Szerokość automatyczna	1/3 szerokości kontenera

Uwaga: 1/3 szerokości to `.col-4`, ponieważ `12 * 1/3 = 4` (mamy 12 kolumn bazowych)

Ustalamy szerokość wszystkich kolumn

Możemy też z góry ustalić szerokość wszystkich kolumn – pod warunkiem, że będą one pasowały do 12-kolumnowej siatki
Bootstrapa.

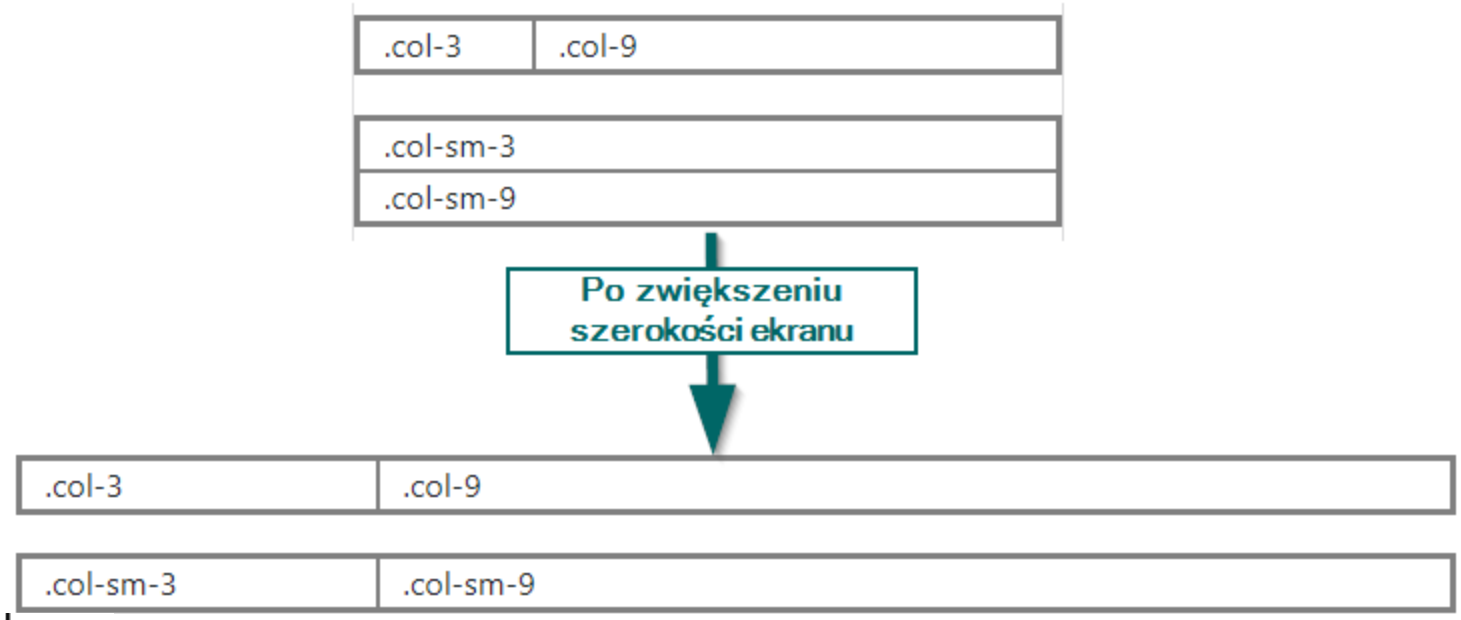
```
<div class="container">
  <div class="row">
    <div class="col-3">1/4 szerokości kontenera</div>
    <div class="col-9">3/4 szerokości kontenera</div>
  </div>
</div>
```

- `.col-3` --> `12 * 1/4 = 3`;
- `.col-9` --> `12 * 3/4 = 9`;

Składanie kolumn na wybranych rozmiarach ekranu

Dzięki użyciu breakpointu (`-sm` , `-md` itp.) możemy sprawić, by kolumny zaczęły się składać po osiągnięciu wybranej
szerokości ekranu.

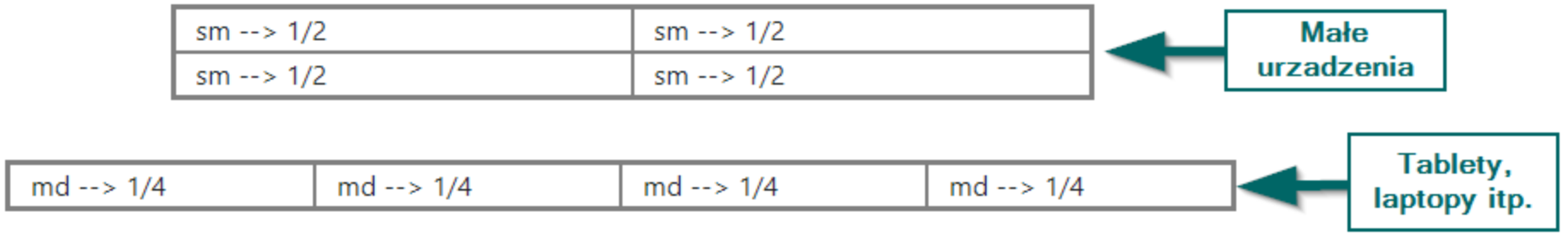
Np. `col-sm-*` zaczynają ułożone jedno na drugim, a na ekranach większych niż `sm` układają się w poziomie. Dla
porównania `.col` nie ma takich właściwości.



Łączenie klas – różne zachowania na różnych urządzeniach

Możemy ustalić inny styl dla małych urządzeń oraz inny dla większych (np. `md` i więcej) poprzez dodanie kilku klas do
elementu.

```
<div class="container">
  <div class="row">
    <div class="col-6 col-md-3">do `md` --> 1/2, `md` i szerzej --> 1/4</div>
    <div class="col-6 col-md-3">do `md` --> 1/2, `md` i szerzej --> 1/4</div>
    <div class="col-6 col-md-3">do `md` --> 1/2, `md` i szerzej --> 1/4</div>
    <div class="col-6 col-md-3">do `md` --> 1/2, `md` i szerzej --> 1/4</div>
  </div>
</div>
```



Obrazki

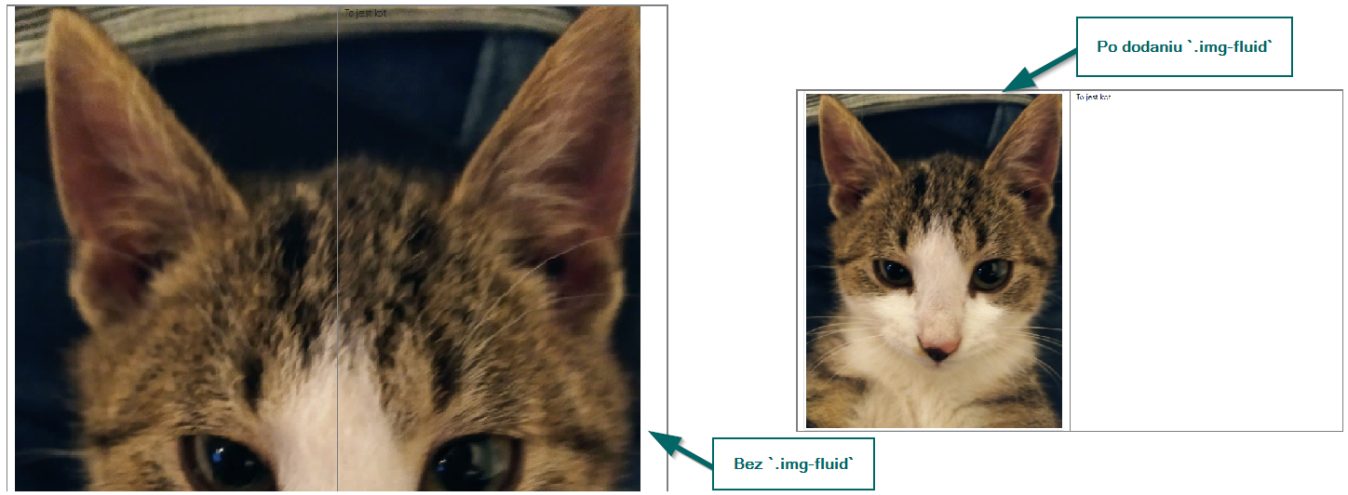
Bootstrap udostępnia klasy sprawiające, że obrazki zachowują się responsywnie oraz pozwalające zmienić ich styl:

- `.img-fluid` – responsywność obrazków;
- `.img-thumbnail` – dodaje ramkę wokół obrazka;
- `float-start`, `float-end` – wyrównanie do lewej / prawej strony;
- `text-center` – wyśrodkowanie.

--> Dokumentacja

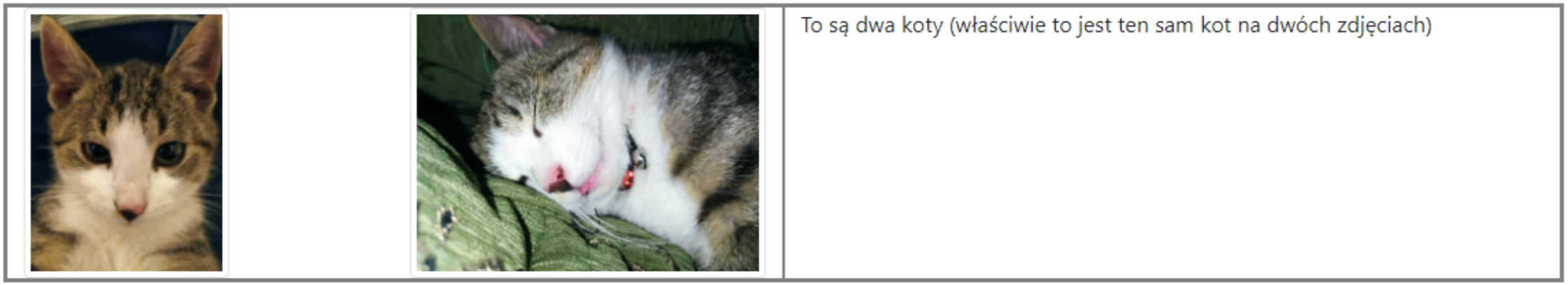
- `img-fluid` sprawi, że obrazek przyjmie szerokość nie większą niż nadrzędny element;
- `img-thumbnail` dodaje zaokrągloną ramkę.

```
<div class="container">
  <div class="row">
    <div class="col-sm-6">
      <img src="../images/unsplash.jpg" class="img-fluid img-thumbnail" />
    </div>
    <div class="col-sm-6">Po lewej mamy ładny obrazek.</div>
  </div>
</div>
```



- `.float-start` oraz `.float-end` pozwolą rozmieścić obrazki wewnątrz kontenera

```
<div class="container">
  <div class="row">
    <div class="col-sm-6">
      <img src="../images/klusek-1.jpg" width="150" class="float-start" />
      <img src="../images/klusek-2.jpg" width="265" class="float-end" />
    </div>
    <div class="col-sm-6">To są dwa koty (właściwie to jest ten sam kot na dwóch zdjęciach)</div>
  </div>
</div>
```



- `.text-center` pozwoli wyśrodkować znajdujący się wewnątrz obrazek

```
<div class="container">
  <div class="row">
    <div class="col-sm-6 text-center">
      <img src="../images/klusek-2.jpg" width="150" />
    </div>
    <div class="col-sm-6">Kot wyśrodkowany</div>
  </div>
</div>
```

Kot wyśrodkowany

Ustalanie marginesów i *paddingu* jest trochę złożone, ale dobrze opisane w dokumentacji. My omówimy sobie marginesy – `m`. Dla _paddingów_ jest tak samo – zamieniamy `m` na `p`.

Klasy zapisujemy używając formatu `m{sides}-{breakpoint}-{size}`, gdzie

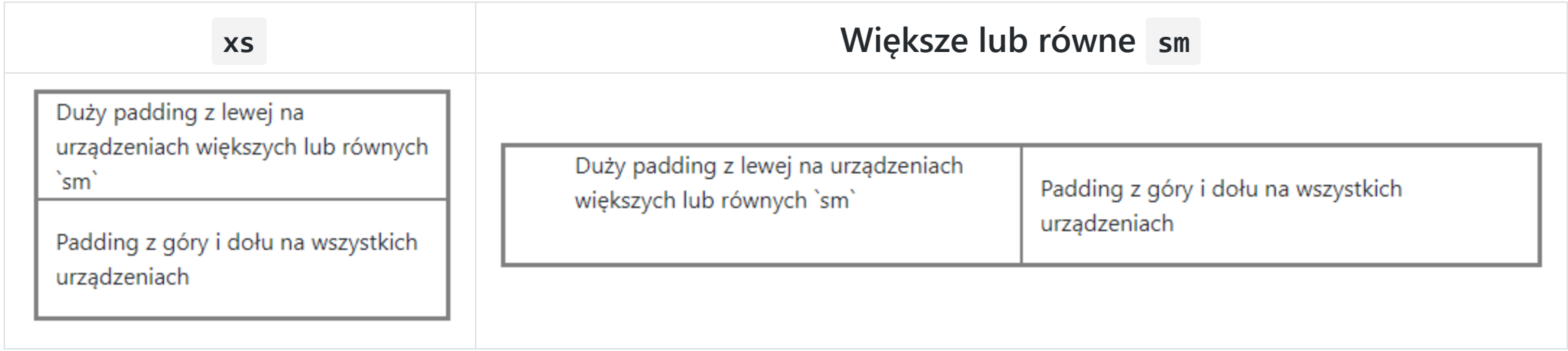
- `sides` to jedną z opcji:
  - `t` – góra (`top`);
  - `b` – dół (`bottom`);
  - `s` – lewy (`start`);
  - `e` – prawy (`end`);
  - `x` – lewy i prawy;
  - `y` – góra i dół;
- `breakpoint` – pusty (jeśli ma działać na wszystkich urządzeniach) lub `s`, `md`, `lg` itd.;
- `size` – cyfra od `0` do `5` (rosnąca wielkość marginesu) lub `auto` (dla `margin-*: auto`).

```
<div class="container">
  <div class="row">
    <div>Jakiś tekst</div>

    <div class="mt-md-3">Margines górny rozmiaru `3` na urządzeniach większych lub równych `md`</div>
  </div>
</div>
```

`xs` lub `sm`	Większe lub równe `md`
Jakiś tekst Margines górny rozmiaru 3 na urządzeniach większych lub równych `md`	Jakiś tekst Margines górny rozmiaru 3 na urządzeniach większych lub równych `md`

- `mt-md-3` – margines górny (`margin-top`) rozmiaru 3 na urządzeniach większych lub równych `md`

```
<div class="container">
  <div class="row">
    <div class="col-sm-6 ps-sm-5">Duży padding z lewej na urządzeniach większych lub równych `sm`</div>
    <div class="col-sm-6 py-3">Padding z góry i dołu na wszystkich urządzeniach</div>
  </div>
</div>
```

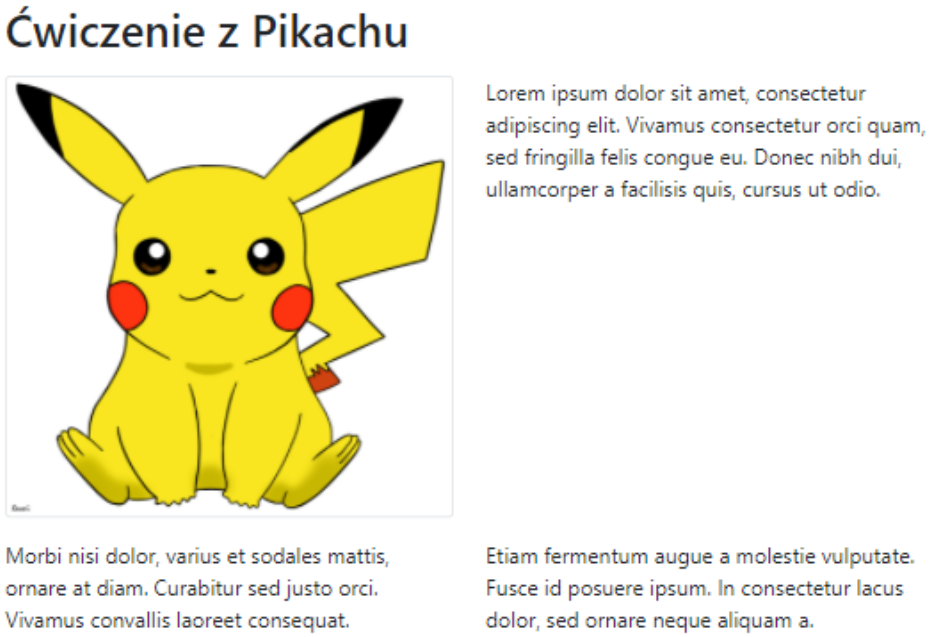


- `ps-sm-5` – *padding* lewy (*start*; `padding-left`) rozmiaru `5` na urządzeniach >= `sm`;
- `py-3` – *padding* górny i dolny (`padding-top`, `padding-bottom`) rozmiaru `3` zawsze.

Ćwiczenie nr 1 ~ 20–30 minut

W poniższym pliku znajdziecie opis ćwiczenia wraz z kodem `html`:

--> https://github.com/pawel-stan/bootstrap-excersise-pikachu/blob/master/pikachu.md



Tabele

Samo dodanie klasy `.table` sprawi, że nasze tabele staną się piękne.

```
<table class="table">
  ...
</table>
```

Bez `.table`	Po dodaniu `.table`

#	Imię	Nazwisko	Kraj
1	Maria	Skłodowska	Polska
2	Roger	Penrose	Wielka Brytania
3	Richard	Feynman	USA

`.table-striped` – co drugi wiersz będzie ciemniejszy

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

#	Imię	Nazwisko	Kraj
1	Maria	Skłodowska	Polska
2	Roger	Penrose	Wielka Brytania
3	Richard	Feynman	USA

`.table-dark` – wersja ciemna

```
<table class="table table-dark">
  ...
</table>
```

#	Imię	Nazwisko	Kraj
1	Maria	Skłodowska	Polska
2	Roger	Penrose	Wielka Brytania
3	Richard	Feynman	USA

`.table-hover` – zaznaczanie wiersza po najechaniu kursorem

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

#	Imię	Nazwisko	Kraj
1	Maria	Skłodowska	Polska
2	Roger	Penrose	Wielka Brytania
3	Richard	Feynman	USA

Ukrywanie elementów – `.d-*-block` oraz `d-*-none`

Bardzo przydatne bywa ukrywanie elementów na wybranych urządzeniach – najczęściej na tych małych.

`.d-{breakpoint}-{value}` – format analogiczny do tego dla marginesów, gdzie `value` to:

- `none` – ukrywa element (`display: none`);
- `block` – ustawia widoczność na `display: block` – odpowiednie dla większości elementów;
- `inline-block` – `display: inline-block` ;
- `table-cell` – `display: table-cell` – dla kolumn tabel (`td` , `th`);
- inne wartości `display` – por. z dokumentacją.

Jeżeli chcemy pokazać jakiś element dopiero na urządzeniach większych lub równych `md`, to:

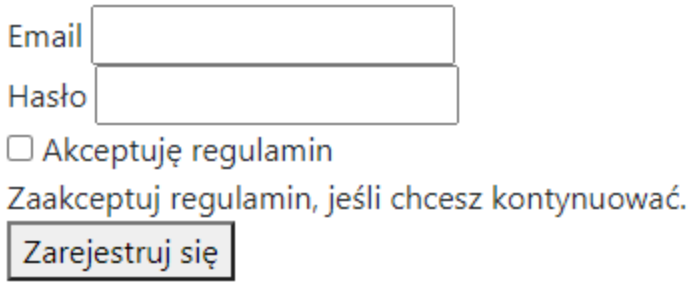
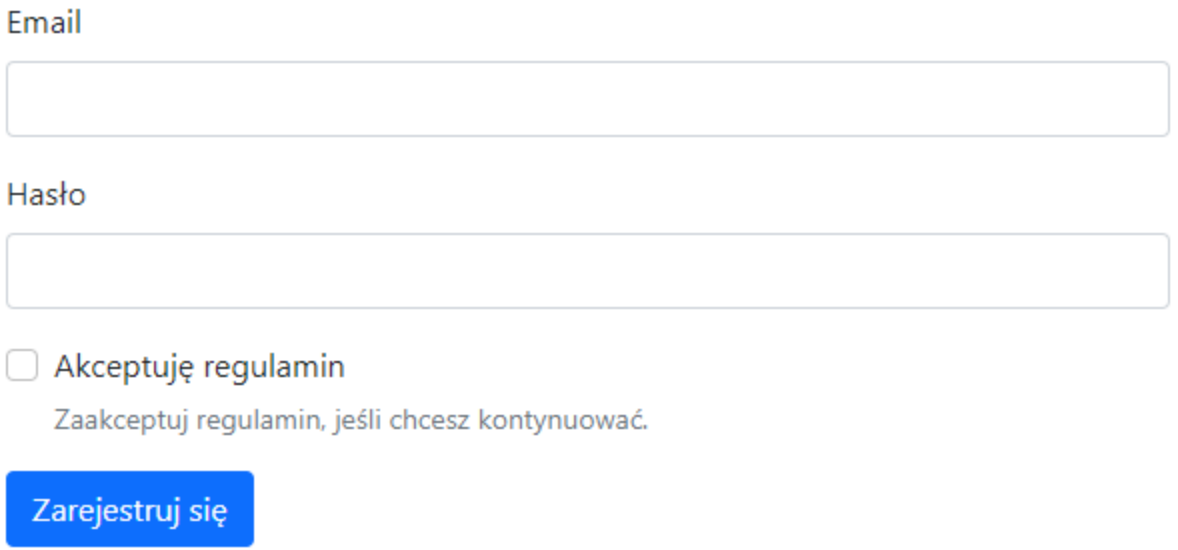
- ukrywamy element całkowicie dodając `.d-none`;
- pokazujemy go dla wybranej wielkości – w tym przypadku `.d-md-block`.

```
<div>Mnie widać zawsze</div>

<div class="d-none d-md-block">Mnie zobaczysz dopiero na `md`</div>
```

`xs`, `sm`	Większe lub równe `md`
Mnie widać zawsze	Mnie widać zawsze Mnie zobaczysz dopiero na `md`

Formularze

Formularze są ważną częścią *Bootstrapa*.

Bez *Bootstrapa*	*Z Bootstrapem*
	

Wersję *bootstrapową* uzsykamy za pomocą poniższego kodu.

```html
<form>
  <div class="mb-3">
    <label for="input-email" class="form-label"> Email </label>
    <input type="email" class="form-control" id="input-email" />
  </div>
  <div class="mb-3">
    <label for="input-password" class="form-label"> Hasło </label>
    <input type="password" class="form-control" id="input-password" />
  </div>
  <div class="mb-3 form-check">
    <input type="checkbox" class="form-check-input" id="check-terms" aria-describedby="terms-help" />
    <label class="form-check-label" for="check-terms"> Akceptuję regulamin </label>
    <div id="terms-help" class="form-text">Zaakceptuj regulamin, jeśli chcesz kontynuować.</div>
  </div>

  <button type="submit" class="btn btn-primary">Zarejestruj się</button>
</form>
```

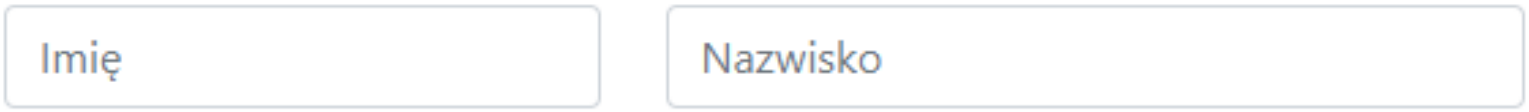
Formularze | Podstawowe klasy

- `.form-label` – etykiety – `label`;
- `.form-control` – pola do wprowadzania danych – `input`, `textarea`;
- `.form-text` – pola opisujące;
- `.form-control-plaintext` – element, który ma się wyświetlić jak zwykły tekst;
- `.form-select` – zmienia wygląd elementu typu `select`;
- `.form-check` + `.form-check-input`, `form-check-label` – zmienia wygląd elementu typu `checkbox`;

Formularze | *Grid system*

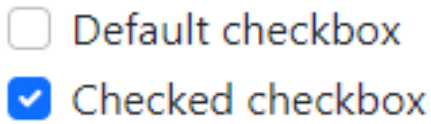
Formularze możemy łaczyć z *grid system* – czyli klasami `row`, `col-*` itd.

```
<form>
  <div class="container">
    <div class="row">
      <div class="col-5">
        <input type="text" class="form-control" placeholder="Imię" />
      </div>
      <div class="col-7">
        <input type="text" class="form-control" placeholder="Nazwisko" />
      </div>
    </div>
  </div>
</form>
```



Formularze | form-check

```
<form>
  <div class="form-check">
    <input class="form-check-input" type="checkbox" value="" id="flexCheckDefault" />
    <label class="form-check-label" for="flexCheckDefault"> Default checkbox </label>
  </div>
  <div class="form-check">
    <input class="form-check-input" type="checkbox" value="" id="flexCheckChecked" checked />
    <label class="form-check-label" for="flexCheckChecked"> Checked checkbox </label>
  </div>
</form>
```



Przykład z dokumentacji.

`input-group`

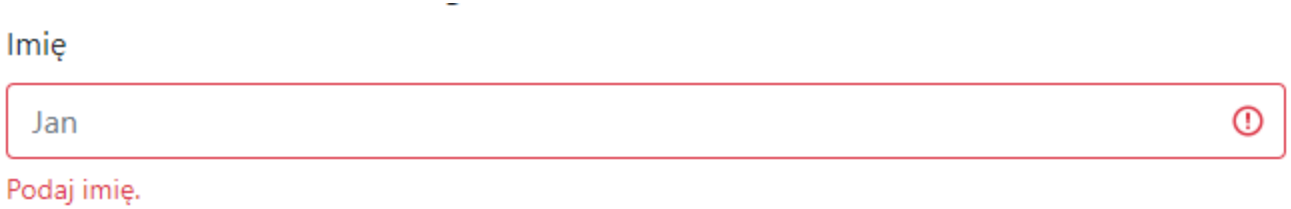
Czasami chcemy wyświetlić jakiś tekst *razem* z polem do wprowadzania danych.

```
<div class="input-group mb-3">
  <span class="input-group-text" id="basic-addon1">@</span>
  <input
    type="text"
    class="form-control"
    placeholder="Username"
    aria-label="Username"
    aria-describedby="basic-addon1"
  />
</div>
```

Przykład z dokumentacji

Dzięki *Bootstrapow* możemy dodać do naszego formularza czytelne komunikaty walidacyjne jeszcze przed wysłaniem danych do serwera – dokumentacja.

```
<form class="needs-validation" novalidate>
  <div class="col-md-6">
    <label for="input-name" class="form-label"> Imię </label>

    <input type="text" class="form-control" id="input-name" placeholder="Jan" required />

    <div class="invalid-feedback">Podaj imię.</div>
  </div>

  <div class="col-12 text-end">
    <button type="submit" class="btn btn-primary">Zarejestruj się</button>
  </div>
</form>
```



Żeby zadziałała walidacji w stylu *Bootstrap*, musimy:

- dodać do formularza atrybut `novalidate`;
- dodać do formularza klasę `needs-validation`;
- opcjonalnie dodać komunikaty walidacyjne dzięki

```
<div class="invalid-feedback">Tekst</div>
```

- zablokować w `javascript` wysłanie formularza do serwera w przypadku błędów;

  Przykładowy kod dostępny jest w dokumentacji.

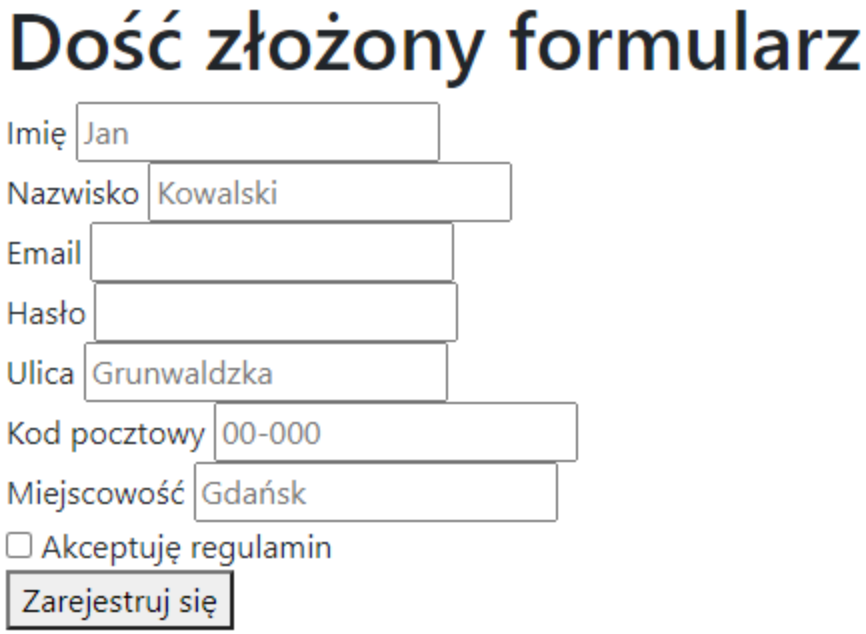
  - `event.preventDefault()` oraz `event.stopPropagation()` odpowiadają za przerwanie wykonywanej operacji – np. wysłania formularza.

Wspomniany kod `javascript` wygląda tak:

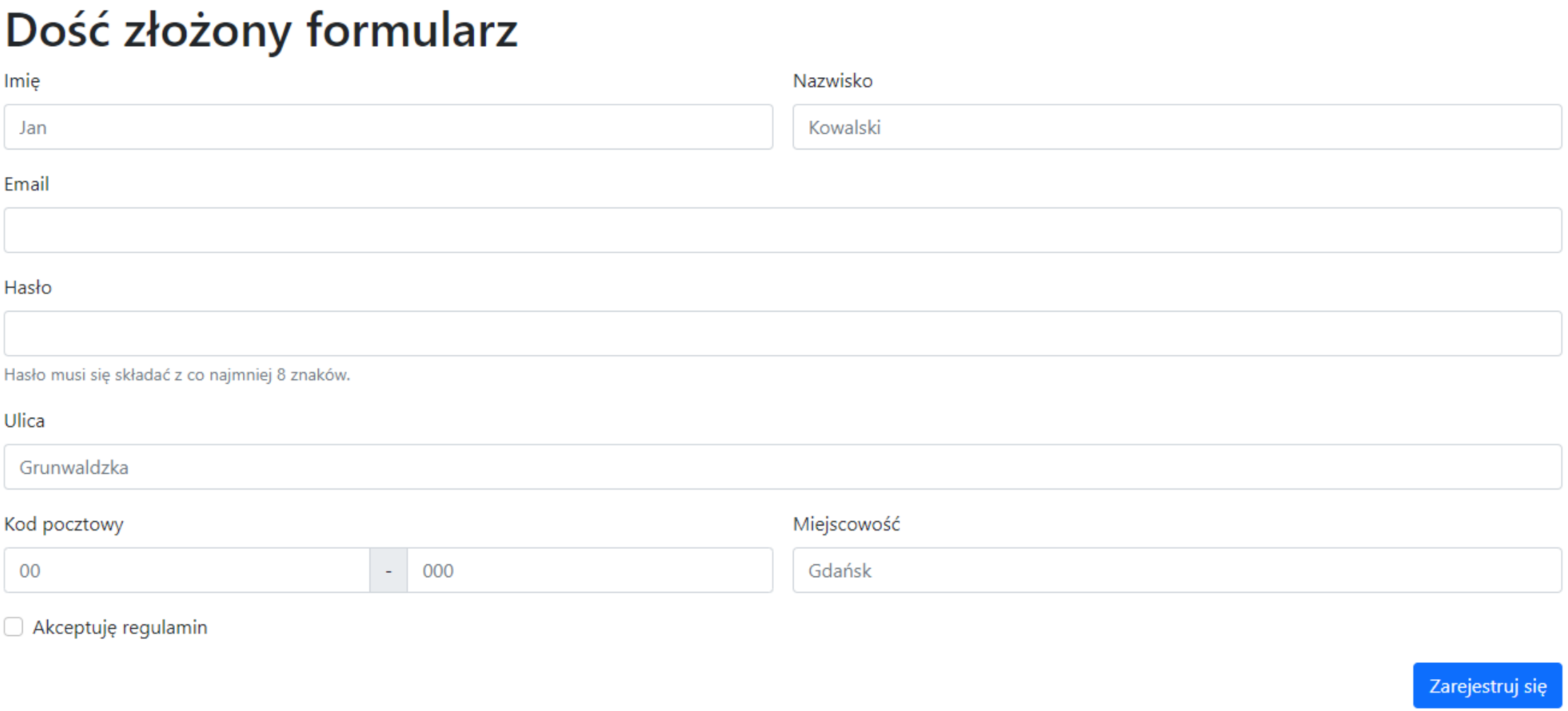
```
(function () {
  "use strict";

  // Fetch all the forms we want to apply custom Bootstrap validation styles to
  var forms = document.querySelectorAll(".needs-validation");

  // Loop over them and prevent submission
  Array.prototype.slice.call(forms).forEach(function (form) {
    form.addEventListener(
      "submit",
      function (event) {
        if (!form.checkValidity()) {
          event.preventDefault();
          event.stopPropagation();
        }

        form.classList.add("was-validated");
      },
      false
    );
  });
})();
```

Ćwiczenie / praca domowa ~ 20–30 minut

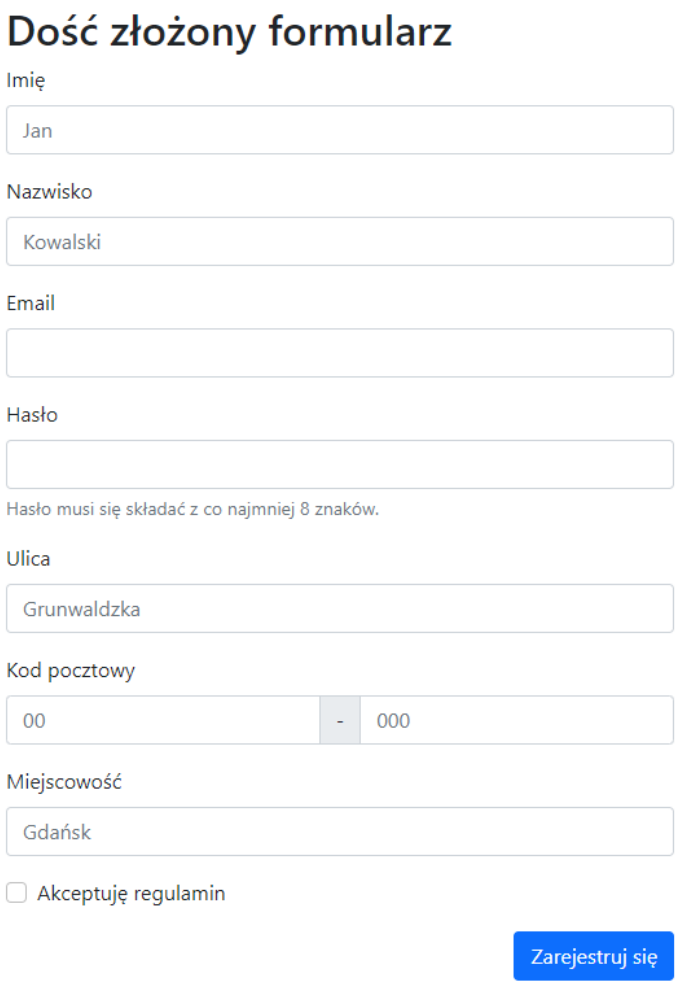
W pliku `form-example-template.html` mamy formularz, który wygląda tak:



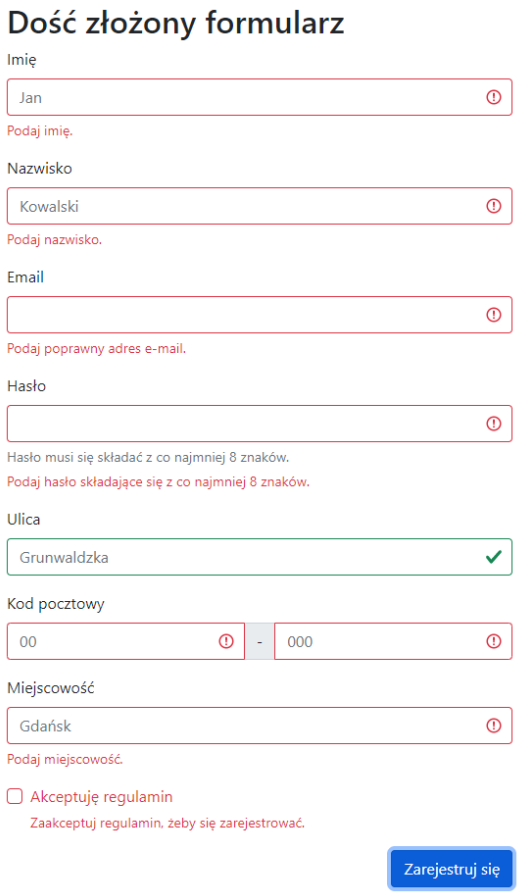
Chcemy, by wyglądał tak na urządzeniach co najmniej `md`:



oraz tak na urządzeniach mniejszych:



Dodatkowo powinna działać walidacja:



Uwaga: Dodałem już potrzebny skrypt `javascript`.

Uwagi:

- tym razem konieczne będzie dodanie kilku elementów (np. tych z komunikatami walidacyjnymi);
- jeżeli komuś tak będzie wygodniej, może usunąć mój formularz i zacząć pisać od zera;
- w większości `input` -ów trzeba dopisać odpowiedni type.