Laboratorium 5: Walidacja formularza

WYMAGANY FORMULARZ Z POPRZEDNICH LABORATORIÓW

Dokładnie prześledź przykład poniżej:

Przykład. Tworzenie funkcji do weryfikacji pól formularza

Walidacja myForm

Język JavaScript jest często używany do wstępnego sprawdzenia poprawności danych wprowadzanych przez użytkowników. Przyjrzyjmy się formularzowi poniżej.

```
<form action="index.html#kontakt" method="post" onsubmit="return checkForm();" >
<fieldset>
	<legend>Formularz kontaktowy</legend>
	<label for="contactName">Imię</label>
	<input type="text" id="contactName"/>

	<label for="contactEmail">Email</label>
	<input type="text" id="contactEmail" />

	<input type="submit" value="Wyślij" />
</fieldset>
</form>
```

Za sprawdzanie poprawności danych odpowiada funkcja *checkForm()*.

Funkcja ta powinna zająć się sprawdzeniem kompletności danych. Odwołując się do właściwości *value* każdego z pól, można stwierdzić, czy zawiera ono jakieś dane, czy też pusty ciąg znaków.

Funkcja *checkForm()* musi zwrócić wartość *true*, jeśli *myForm* ma zostać wysłany (czyli kiedy dane są kompletne), oraz *false* w przeciwnym razie wypadku.

```
<script>
function checkForm()
{
  var error=false; //to znaczy, że danych nie brakuje
  var errorText=""; //komunikat z błędem
  var contactName = document.getElementById("contactName");
  var contactEmail = document.getElementById("contactEmail");
```

// jeśli nic nie wpisano w *contactName* to jest błąd - sprawdzamy czy *contactName* jest puste, jeśli tak to dodajemy do *errorText* pole imię i oznaczamy, że brakuje danych

```
  if (contactName.value == ""){
    errorText += "imię\n"
    error=true;
  }
```

// jeśli nic nie wpisano w *contactEmail* to jest błąd - sprawdzamy czy *contactEmail* jest puste, jeśli tak to dodajemy do *errorText* pole email i oznaczamy, że brakuje danych

```
  if (emailName.value == ""){
    errorText += "email\n"
    error=true;
  }
```

// jeśli nie brakuje danych wysyłamy formularz, jeśli brakuje pojawia się komunikat i formularz nie zostanie wysłany

```
  if (!error) return true;
  else{
    alert ("Nie wypełniłeś następujących pól:\n" + errorText);
    return false;
  }
}
</script>
```

Ostatecznie, jeśli zmienna *error* ma wartość *true*, za pomocą metody *alert* jest wyświetlane okno z błędem, kiedy natomiast ma wartość *false*, jest wykonywana metoda *submit* wysyłająca dane z formularza.

Sprawdzanie poprawności adresu email

W kolejnym kroku możemy sprawdzić, czy podany adres email jest poprawny. W tym celu modyfikujemy tylko tę część:

```
if(contactEmail.value == ""){
    errorText += "email\n";
    error=true;
}
else
{
    var email = contactEmail.value;
    var regex = /^[a-zA-Z0-9._-]+@([a-zA-Z0-9.-]+\.)+[a-zA-Z0-9.-
]{2,4}$/;
    if(regex.test(email)==false)
    {
       errorText += "błędny email\n";
       error=true;
    }
}
```

Ćwiczenie 2. Walidacja formularza

1. Utwórz nowy plik forms.js i zapisz go w katalogu js.
2. Dodaj ścieżkę do pliku na dole strony index.html (tak jak jquery).
3. Korzystając z przykładu powyżej napisz funkcję sprawdzającą pola formularza, możesz skopiować kod z przykładu i dodać do niego pozostałe pola.
4. Zmodyfikuj skrypt funkcji sprawdzającej formularz w taki sposób, żeby zamiast komunikatu o błędach wyświetlał ukryte akapity. W tym celu usuń komunikat o błędach z końca skryptu, natomiast w poszczególnych warunkach sprawdzających czy elementy są właściwie wypełnione zamiast *errorText* wstaw dodawanie odpowiedniej klasy z błędem.

   Na przykład:

```
if (contactEmail.value == ""){
    errorText += "email\n";
    error=true;
}
```

zamień na:

```
if(contactEmail.value == ""){
    document.getElementById('contactEmail').className='form-
    control is-invalid';
    document.getElementById('errorEmail').innerHTML='Adres email
    jest wymagany';
    error=true;
} else {
    document.getElementById('contactEmail').className='form-
    control is-valid';
 }
```

5. W ukrytych akapitach z błędami podmień klasę **d-none** na **invalid-feedback**.
6. Wprowadź walidację poprawności adresu email i zmień komunikat o błędzie, jeśli adres jest podany, ale w złym formacie (innerHTML).

```
if(contactEmail.value == ""){
    document.getElementById('contactEmail').className='form-
    control is-invalid';
    document.getElementById('errorEmail').innerHTML='Adres email
    jest wymagany';
    error=true;
} else {
    var email = contactEmail.value;
    var regex = /^[a-zA-Z0-9._-]+@([a-zA-Z0-9.-]+\.)+[a-zA-Z0-9.-
]{2,4}$/;
    if(regex.test(email)==false) {
        document.getElementById('contactEmail').className='form-
        control is-invalid';
        document.getElementById('errorEmail').innerHTML='Podany
        adres email jest nieprawidłowy';
        error=true;
    }
 } else {
    document.getElementById('contactEmail').className='form-
    control is-valid';
  }
```

Ćwiczenie 3. Podświetlanie kontrolek

1. Korzystając z podpowiedzi na https://getbootstrap.com/docs/4.0/components/forms/#server-side zmodyfikuj skrypt w taki sposób, żeby w momencie, gdy formularz jest wysyłany z błędem, kontrolki formularza otrzymywały odpowiednie podświetlenie – zielone dla pól poprawnie podświetlonych i czerwone dla niepoprawnych.

Zadanie samodzielne

1. Wykorzystaj zdarzenie *onchange* lub *onblur* oraz wiedzę z przykładów powyżej i napisz funkcje pomocnicze, które będą sprawdzały wartości w polach formularza od razu po ich wypełnieniu (a nie podczas wysyłania formularza) i stosowały właściwe podświetlenie oraz komunikat o błędzie.

Podpowiedź:

1. Dodaj do pola <input> zdarzenie onBlur i przypisz mu funkcję, np. checkEmail().
2. W forms.js utwórz nową funkcję checkEmail() i skopiuj do niej część sprawdzającą poprawność adresu email.
3. Usuń z niej error=’true’.