

Podstawy informatyki

HTML5 / CSS3

Laboratorium 3

Formularze HTML: pola, etykiety i przesyłanie danych

Czas trwania: 2 × 45 min**Tryb:** studia stacjonarne**Jednostka:** Katedra Elektrotechniki i Technologii Nadprzewodnikowych**Prowadzący:** dr inż. Michał Łanczont

Cel ćwiczenia

Celem jest zbudowanie poprawnego i dostępnego formularza, poznanie atrybutów **action**, **method**, **name**, **value**, typów pól, etykiet, grupowania przez **fieldset/legend** oraz podstaw walidacji po stronie przeglądarki.

Wstęp

Formularz HTML służy do zbierania danych od użytkownika. Sam HTML nie zapisuje danych w bazie ani nie wysyła wiadomości e-mail; opisuje interfejs i sposób utworzenia żądania. Przetwarzanie danych wykonuje serwer albo kod JavaScript.

Element form: action i method

```
<form action="/odbierz-dane" method="get">
  ...
</form>
```

Atrybut **action** określa adres docelowy. Jeżeli go pominiemy, formularz jest wysyłany do bieżącego adresu dokumentu. **method="get"** dołącza dane do URL, natomiast **method="post"** umieszcza je w ciele żądania.

GET/POST i bezpieczeństwo

POST nie oznacza automatycznie, że dane są zaszyfrowane lub “ukryte” przed podsłuchem. Nie są one dopisywane do paska adresu, ale poufność transmisji zapewnia HTTPS. Hasła i innych sekretów nie należy przysyłać metodą GET, ponieważ mogą znaleźć się w historii, logach i linkach.

Użycie **action="#"** nie zatrzymuje domyślnej wysyłki formularza. Jeżeli formularz ma być obsługiwany tylko w JavaScript, zdarzenie **submit** musi zostać przechwycone, np. przez **preventDefault()**. W tym kursie JavaScript nie jest wymagany, więc do obserwacji parametrów najlepiej wykorzystać GET i prosty adres docelowy.

Najczęstsze typy input

Typ	Zastosowanie	Przykład
text	pojedynczy wiersz tekstu	<code><input type="text" name="imie"></code>
password	maskowanie wpisywanego tekstu	<code><input type="password" name="haslo"></code>
email	pole z walidacją składni adresu e-mail	<code><input type="email" name="email"></code>
number	wartość liczbowa z ograniczeniami min/max/step	<code><input type="number"></code>
date	data, z interfejsem zależnym od przeglądarki	<code><input type="date" name="data"></code>
color	wybór koloru	<code><input type="color" name="kolor"></code>
range	suwak zakresu	<code><input type="range" min="0" max="10"></code>
checkbox	niezależna opcja wielokrotnego wyboru	<code><input type="checkbox" ...></code>
radio	jeden wybór w grupie o wspólnym name	<code><input type="radio" ...></code>

Pole **number** reprezentuje liczbę, ale nie należy upraszczać jego opisu do “tylko cyfr”: dopuszczalne wartości zależą m.in. od **step**, lokalizacji i walidacji przeglądarki.

label, id, name i value

label powinien być programowo powiązany z kontrolką. Najczęściej realizuje się to przez parę **for/id**:

```
<label for="user-name">Imie:</label>
<input type="text" id="user-name" name="imie" required>
```

Atrybut **name** jest kluczem wysyłanym razem z wartością pola. Kontrolka bez **name** co do zasady nie wnosi swojej wartości do zwykłego zestawu danych formularza.

textarea, select i button

```
<label for="uwagi">Twoje uwagi:</label>
<textarea id="uwagi" name="uwagi" rows="4"></textarea>

<button type="submit">Wyslij</button>
<button type="reset">Wyczysc</button>
<button type="button">Przycisk bez wysylki</button>
```

W formularzu brak jawnego **type** w **button** może oznaczać przycisk wysyłający, dlatego w przykładach warto podawać typ jawnie.

Radio i checkbox

Radio należą do jednej grupy, jeżeli mają wspólny **name**:

```
<label><input type="radio" name="rola" value="student"> Student</label>
<label><input type="radio" name="rola" value="pracownik"> Pracownik</label>
<label><input type="radio" name="rola" value="inna"> Inna rola</label>
```

Checkboxy opisujące jedną kategorię mogą również używać wspólnego **name** i różnych wartości. Po zaznaczeniu kilku pozycji w query string pojawi się ten sam klucz wielokrotnie.

```
<label><input type="checkbox" name="technologia" value="html"> HTML</label>
<label><input type="checkbox" name="technologia" value="css"> CSS</label>
<label><input type="checkbox" name="technologia" value="js"> JavaScript</label>
```

Walidacja wbudowana w HTML

Atrybuty `required`, `min`, `max`, `minlength`, `maxlength`, `pattern` i odpowiedni `type` mogą włączyć walidację po stronie przeglądarki. Jest to wygodne dla użytkownika, ale nie zastępuje walidacji po stronie serwera.

Obserwacja GET

```
<form action="wynik.html" method="get">
  <label for="username">Nazwa:</label>
  <input id="username" name="uzytkownik" required>

  <label for="age">Wiek:</label>
  <input type="number" id="age" name="wiek" value="30">

  <button type="submit">WYSLIJ</button>
</form>
```

Po wysłaniu można otrzymać URL podobny do:

```
wynik.html?uzytkownik=Anna&wiek=30
```

Znak `?` rozpoczyna część zapytania, a poszczególne pary są kodowane w formacie właściwym dla URL. Długość praktycznie użytecznego adresu jest ograniczona przez przeglądarki, serwery i pośredników, dlatego GET nie służy do przesyłania dużych formularzy.

Przykład kompletny – formularz obserwowany przez GET

```
<form action="wynik.html" method="get">
  <label for="imie">Imie:</label>
  <input id="imie" name="imie" required>

  <fieldset>
    <legend>Rola</legend>
    <label><input type="radio" name="rola" value="student" required> Student</label>
    <label><input type="radio" name="rola" value="pracownik"> Pracownik</label>
  </fieldset>

  <fieldset>
    <legend>Technologie</legend>
    <label><input type="checkbox" name="technologia" value="html"> HTML</label>
    <label><input type="checkbox" name="technologia" value="css"> CSS</label>
  </fieldset>

  <label for="uwagi">Uwagi:</label>
  <textarea id="uwagi" name="uwagi" rows="4"></textarea>

  <button type="submit">Wyslij</button>
  <button type="reset">Wyczysc</button>
</form>
```

`fieldset` i `legend` grupują logicznie powiązane kontrolki i poprawiają zrozumiałość formularza. Po wysłaniu dwóch checkboxów o wspólnym `name` parametr `technologia` pojawi się więcej niż raz.

Typowe błędy formularzy

- etykieta `for` wskazuje ID, które nie istnieje,

- dwa różne pola mają to samo ID,
- radio w jednej grupie mają różne **name** i dlatego można zaznaczyć kilka,
- pole wygląda poprawnie, ale nie ma **name** i nie jest wysyłane,
- przesyłanie hasła metodą GET,
- utożsamianie POST z szyfrowaniem.

Przebieg ćwiczenia

Zadanie 1. Prosty formularz kontaktowy

Do wykonania

Utwórz `kontakt.html` z pełnym szkieletem HTML. Dodaj formularz `method="get"` prowadzący do lokalnego `wynik.html`. Dodaj pola Imię i Email, poprawne **name**, unikalne **id** i powiązane **label**.

Zadanie 2. Formularz rejestracyjny i required

Do wykonania

Dodaj pole hasła. Pola imienia, e-mail i hasła oznacz jako wymagane. Dodaj jawny **button** `type="submit"`. Nie wysyłaj w tym ćwiczeniu rzeczywistego hasła do zewnętrznej usługi; pole służy wyłącznie do nauki struktury formularza.

Zadanie 3. Radio buttons

Do wykonania

Dodaj pytanie “Jaka jest Twoja rola?” z trzema opcjami radio: Student, Pracownik, Inna. Wszystkie kontrolki mają mieć wspólny **name="rola"**, ale różne **value**. Każda opcja ma posiadać etykietę.

Zadanie 4. Checkboxy, textarea i reset

Do wykonania

Dodaj grupę “Jakie technologie znasz?” z checkboxami o wspólnym **name="technologia"** i różnych wartościach. Dodaj **textarea** z `rows="4"`, a obok przycisku wysyłania przycisk `type="reset"`.

Zadanie 5. Pola specjalne i ograniczenia

Do wykonania

Dodaj wiek jako `type="number"` z `min="18"` i sensownym `max`; dodaj datę wizyty, kolor i suwak oceny z `min`, `max` oraz `step`. Sprawdź w kilku kontrolkach działanie **required**.

Zadanie 6. Analiza przesłanych danych

Do wykonania

Wypełnij formularz, zaznacz jedną opcję radio i dwa checkboxy. Wyślij GET i przeanalizuj query string. Wypisz w komentarzu HTML pary `name=value`, które pojawiły się w URL. Sprawdź, co dzieje się z polem bez atrybutu `name`.

Zadanie 7. Porównanie GET i POST

Do wykonania

Utwórz kopię formularza z `method="post"`. Jeżeli używasz wyłącznie statycznych plików i nie masz serwera obsługującego POST, obserwuj w DevTools próbę wysłania żądania i porównaj ją z GET. Wyjaśnij, dlaczego brak parametrów w pasku adresu nie oznacza szyfrowania danych.

Zadanie końcowe

Integracja

Zbuduj kompletny formularz zgłoszeniowy z co najmniej sześcioma różnymi typami kontrolek, etykietami, `name`, grupą radio, grupą checkboxów, `textarea`, walidacją HTML oraz przyciskami submit/reset. Wersja GET ma umożliwić analizę danych w URL.

Kryteria zaliczenia

- poprawne etykiety i atrybuty `name`,
- właściwe grupowanie radio/checkbox,
- poprawne rozróżnienie GET i POST,
- brak przesyłania poufnych danych przez GET,
- wykorzystanie `required`, `min/max/step` w uzasadniony sposób.

Sprawozdanie

Do odbioru należy przygotować pliki źródłowe utworzone podczas ćwiczenia. Jeżeli prowadzący wymaga sprawozdania, powinno ono zawierać: krótki opis rozwiązania, najważniejsze fragmenty kodu, zrzut ekranu wyniku oraz wnioski dotyczące napotkanych problemów. Pliki HTML i CSS należy zachować w strukturze katalogów zgodnej z instrukcją.

Materiały pomocnicze

- HTML Living Standard: <https://html.spec.whatwg.org/>
- MDN Web Docs – HTML: <https://developer.mozilla.org/docs/Web/HTML>
- MDN Web Docs – CSS: <https://developer.mozilla.org/docs/Web/CSS>
- Walidator HTML: <https://validator.w3.org/>