



Zintegrowany  
Program  
Rozwoju  
Politechniki  
Lubelskiej -  
część druga

**POLITECHNIKA LUBELSKA  
WYDZIAŁ ELEKTROTECHNIKI  
I INFORMATYKI**

**KIERUNEK STUDIÓW  
INFORMATYKA**

*MATERIAŁY DO ZAJĘĆ  
LABORATORYJNYCH*

Podstawy aplikacji internetowych

Autor:  
dr Beata Pańczyk

Lublin 2021



Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska

Unia Europejska  
Europejski Fundusz Społeczny



## INFORMACJA O PRZEDMIOCIE

### Cele przedmiotu:

Cel 1. Zapoznanie studentów z podstawowymi zagadnieniami tworzenia aplikacji internetowych z wykorzystaniem aktualnych standardów projektowania i programowania obowiązującymi w Internecie.

Cel 2. Nabycie przez studentów umiejętności tworzenia prostych aplikacji internetowych z wykorzystaniem współczesnych technologii i narzędzi.

### Efekty kształcenia w zakresie umiejętności:

Efekt 1. Student umie tworzyć proste aplikacje internetowe w wybranych środowiskach programistycznych stosując poznane technologie i narzędzia.

Efekt 2. Student potrafi ocenić przydatność technologii służących do tworzenia prostych aplikacji internetowych oraz wybierać i stosować właściwe metody i narzędzia do realizacji zadanej funkcjonalności aplikacji webowej.

### Literatura do zajęć:

#### Literatura podstawowa

1. Frahaan Hussain, Responsive Web Design. Nowoczesne strony WWW na przykładach, Helion 2019.
2. <http://www.w3schools.com> (kursy technologii internetowych).

#### Literatura uzupełniająca

1. Ochim H., Pańczyk B., RWD jako narzędzie optymalizacji stron internetowych, Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Środowiska, 2016.

### Metody i kryteria oceny:

#### Oceny cząstkowe:

- Ocena 1 Przygotowanie merytoryczne do zajęć laboratoryjnych na podstawie: wykładów, literatury, pytań kontrolnych do zajęć.
- Ocena 2 Realizacja i prezentacja projektu cząstkowego.

#### Ocena końcowa - zaliczenie przedmiotu:

- Pozytywne oceny cząstkowe.
- Prezentacja projektu końcowego aplikacji internetowej.

**Plan zajęć laboratoryjnych:**

<a href="#"><u>Lab1.</u></a>	Edytor tekstowy HTML i środowisko programowania prostych aplikacji internetowych. Lokalny serwer HTTP. Budowa dokumentu HTML. Walidacja kodu HTML z wykorzystaniem validatora W3C. Tworzenie graficznego interfejsu użytkownika na bazie podstawowych elementów formularzy HTML.
<a href="#"><u>Lab2.</u></a>	Wykorzystanie podstawowych arkuszy CSS do formatowania strony wizualnej dokumentu HTML. Walidacja kodu dokumentów HTML i CSS.
<a href="#"><u>Lab3.</u></a>	Pozycjonowanie elementów na stronie internetowej. Wykorzystanie narzędzi deweloperskich dostępnych w przeglądarce internetowej do kontroli arkuszy CSS.
<a href="#"><u>Lab4.</u></a>	Poznanie zaawansowanych możliwości tworzenia formularzy HTML5. Realizacja walidacji danych w polach formularzy. Metody przesyłania danych z formularza po stronie klienta na serwer WWW – wykorzystanie narzędzi dostępnych w przeglądarce. Wsparcie przeglądarek dla pól formularzy HTML5.
<a href="#"><u>Lab5.</u></a>	Realizacja responsywnej strony internetowej z wykorzystaniem zapytania o media.
<a href="#"><u>Lab6.</u></a>	Wprowadzenie elementów interakcji z użytkownikiem w aplikacji internetowej za pomocą języka skryptowego JavaScript. Realizacja obsługi zdarzeń na stronie WWW z zastosowaniem modelu DOM. Korzystanie z konsoli JavaScript dostępnej w przeglądarce internetowej.
<a href="#"><u>Lab7.</u></a>	Budowa prostej aplikacji typu SPA z wykorzystaniem funkcji JavaScript.
<a href="#"><u>Lab8.</u></a>	Obsługa i walidacja formularzy po stronie klienta za pomocą JavaScript.
<a href="#"><u>Lab9.</u></a>	Zastosowania biblioteki jQuery w aplikacji internetowej.
<a href="#"><u>Lab10</u></a>	Tworzenie responsywnego szablonu strony z zastosowaniem szkieletu Bootstrap.
<a href="#"><u>Lab11.</u></a>	Budowa aplikacji wykorzystującej geolokację i lokalne magazyny danych.
<a href="#"><u>Lab12.</u></a>	Wykorzystanie elementów programowania obiektowego w JavaScript. Realizacja aplikacji pracującej z danymi w formacie JSON.
<a href="#"><u>Lab13.</u></a>	Prezentacja projektów.



## **LABORATORIUM 1. CZEŚĆ 1. EDYTOR TEKSTOWY HTML I ŚRODOWISKO PROGRAMOWANIA PROSTYCH APLIKACJI INTERNETOWYCH. LOKALNY SERWER HTTP. BUDOWA DOKUMENTU HTML. WALIDACJA KODU HTML Z WYKORZYSTANIEM WALIDATORA W3C.**

### **Cel laboratorium:**

Celem zajęć jest przygotowanie dokumentu HTML zgodnego ze standardami W3C, który będzie stanowił podstawę do realizacji kolejnych zajęć laboratoryjnych.

### **Zakres tematyczny zajęć:**

Wybór edytora i środowiska programistycznego do tworzenia prostych aplikacji internetowych.

Budowa dokumentu HTML5 z zastosowaniem podstawowych znaczników.

Walidacja kodu HTML za pomocą validatora W3C.

### **Pytania kontrolne:**

1. Jakie znaczniki (tagi) powinny się znajdować w bloku <**head**> dokumentu HTML5?
2. Co to są elementy blokowe i liniowe? Podaj przykłady.
3. Co to są elementy z zawartością i bez zawartości? Podaj przykłady.
4. Co to są atrybuty elementów i jak należy je definiować? Podaj przykłady.

### **Zadanie 1.1. Podstawowa struktura dokumentu HTML5**

Utwórz dokument HTML5 postaci jak na listingu 1.1. Wykorzystaj w tym celu np. edytor Notepad++ (wybierz *Language->HTML* oraz *Encoding->Encode in UTF-8 without BOM*).

Dokument HTML składa się zasadniczo z:

- definicji typu dokumentu (**DOCTYPE**),
- właściwego dokumentu zawartego pomiędzy znacznikiem <**html**> ... </**html**>.

Właściwy dokument HTML zawiera zawsze dwa znaczniki (nazywane też tagami lub elementami) zagnieżdżone wewnętrz głównego elementu <**html**>:

- <**head**> ... </**head**> - część nagłówkową z informacjami dla przeglądarki, która zawiera m.in. znaczniki <**meta**> (tzw. metadane), <**title**> (tytuł strony wyświetlany w pasku tytułu okna przeglądarki), <**link**> (będzie nam niezbędny do wskazania pliku z arkuszem CSS opisującym sposób formatowania poszczególnych elementów na stronie), <**script**> (będziemy wykorzystywać w celu dodania kodu JavaScript).
- <**body**> ... </**body**> - właściwą treść (ciało) dokumentu widoczną w oknie przeglądarki.

Rozróżnia się dwa rodzaje znaczników:

- znaczniki parzyste (np. **body**, **head**, **div**, **p**) z zawartością postaci: <**znacznik**> **zawartość** </**znacznik**>
- znaczniki puste (bez zawartości, np. **img**, **br**, **input**) postaci np.: <**znacznik** />, przy czym w standardzie HTML5 znak / jest opcjonalny.



**Fundusze Europejskie**  
Wiedza Edukacja Rozwój



**Rzeczpospolita Polska**



**Unia Europejska**  
Europejski Fundusz Społeczny

Zauważ, że wszystkie znaczniki pisane są małymi literami. Każdy ze znaczników może posiadać różne dodatkowe atrybuty (właściwości, parametry) np.:

<znacznik par1="wartość\_par1" par2="wartość\_par2" ... >

Nazwy parametrów również pisane są małymi literami, dodatkowo każdy parametr, po znaku równości posiada wartość ujętą w znaki cudzysłowy " " lub proste apostrofy ''.

W znaczniku <body> zagnieżdżane są zwykle kolejne elementy blokowe (np. <div>), umożliwiające budowę struktury strony, łatwą do późniejszego opisu strony wizualnej za pomocą reguł CSS. Każdy taki element może (*nie musi*) posiadać atrybut **id**. Atrybut **id** nie jest obowiązkowy, ale jeśli występuje, jego wartość powinna być unikatowa. Na listingu 1.1 zdefiniowano 5 przykładowych elementów <div> - jeden nadrzędny o wartości **id="kontener"** i cztery w nim zagnieżdżone, które wykorzystamy do ustalenia struktury budowanej strony WWW.

### **Listing 1.1. Przykładowa struktura dokumentu HTML5 z blokami div**

```
<!DOCTYPE html>
<html lang="pl">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title> Szybki kurs HTML </title>
</head>
<body>
    <div id="kontener">
        <div id="baner"></div>
        <div id="menu"></div>
        <div id="tresc"></div>
        <div id="stopka"></div>
    </div>
</body>
</html>
```

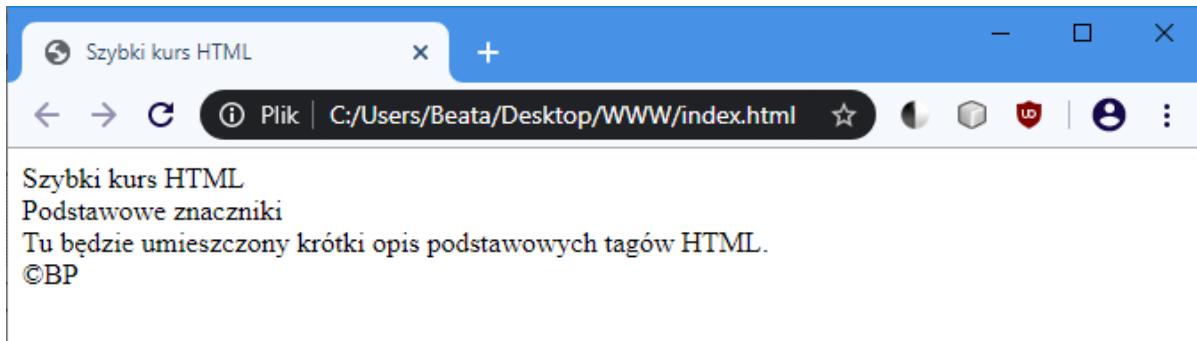
Na listingu 1.2 przedstawiono podstawową strukturę dokumentu HTML. Już teraz warto zwrócić uwagę na atrybut **viewport**, wykorzystywany do wyśrodkowania zawartości na ekranie. Mechanizm jest odpowiednikiem operacji zbliżania za pomocą gestu przybliżenia (ang. pinch-to-zoom), stosowanego w przypadku stron, które nie są przystosowane do urządzeń mobilnych.

Kolejny etap to dodanie zawartości do przygotowanych bloków <div>. Listing 1.2 przedstawia bloki uzupełnione przykładową treścią. W celu wstawienia pewnych znaków specjalnych stosowane są odpowiednie encje – np. znak © wstawiono jako &copy;.

### **Listing 1.2. Bloki div z zawartością**

```
<body>
<div id="kontener">
    <div id="baner">Szybki kurs HTML</div>
    <div id="menu">Podstawowe znaczniki</div>
    <div id="tresc">Tu będzie umieszczony krótki opis podstawowych
        tagów HTML.
    </div>
    <div id="stopka"> &copy;BP </div>
</div>
</body>
```

Utwórz nowy folder na pliki naszego projektu (o nazwie np. WWW) i gotowy dokument zapisz w tym folderze pod nazwą *index.html*. Po otwarciu pliku w przeglądarce otrzymamy obraz podobny do tego na rys.1.1.



Rys.1.1. Plik z listingu 1.2 zinterpretowany w przeglądarce

### Zadanie 1.2. Dodawanie zawartości do elementów blokowych <div>

- Skorzystaj z uzupełniających materiałów wykładowych i kursu <https://www.w3schools.com/> (sprawdź jakie działanie mają znaczniki `<h1>`, `<h2>`, `<p>`, `<br />`, `<ul>`, `<ol>`, `<li>`, `<img>` i `<a>`) i zmień zawartość elementu o `id="baner"` (wykorzystaj znacznik np. `<h2>`) oraz `id="trecs"` (wykorzystaj znaczniki `<ol>` lub `<ul>` i `<li>`), tak aby uzyskać obraz widoczny na rys. 1.2.

#### Wskazówki

- Znacznik `<h2>` można wykorzystać zagnieźdzając go w bloku `<div>` np.: `<div id="baner"> <h2>Szybki kurs HTML</h2> </div>`
  - Ponieważ znaki `<`, `>` pełnią funkcję znaków sterujących w HTML, to umieszczenie takich znaków w treści strony jest możliwe po zastosowaniu odpowiednich encji (zamienników) np. `&lt;html&gt;` - encja `&lt;` oznacza znak mniejszości, `&gt;` - znak większości. Encja **zawsze** rozpoczyna się znakiem `&` i kończy znakiem średnika.
- Na końcu zawartości elementu div o `id="trecs"` wstaw znacznik akapitu `<p> ... </p>`, w którym należy umieścić napis: **HTML validator** i obok obrazek pobrany ze strony: <http://validator.w3.org/>.
  - Dodaj hiperłącze do wskazanej strony validatora. Sprawdź jak stosuje się znacznik `<a>`. W naszym przypadku hiperłączem powinien być fragment (łącznie z obrazkiem):



- Wstaw również hiperłącze do treści **Podstawowe znaczniki** w elemencie o `id="menu"`. Link ma się odnosić do pliku lokalnego *index.html* (czyli do dokumentu bieżącego). Obok niego umieść kolejne hiperłącza: **Tworzenie tabel** i **Budowa formularzy**, odpowiednio do plików (które utworzysz w następnych ćwiczeniach) *tabele.html* i *formularze.html*. Sprawdź działanie hiperłączy.



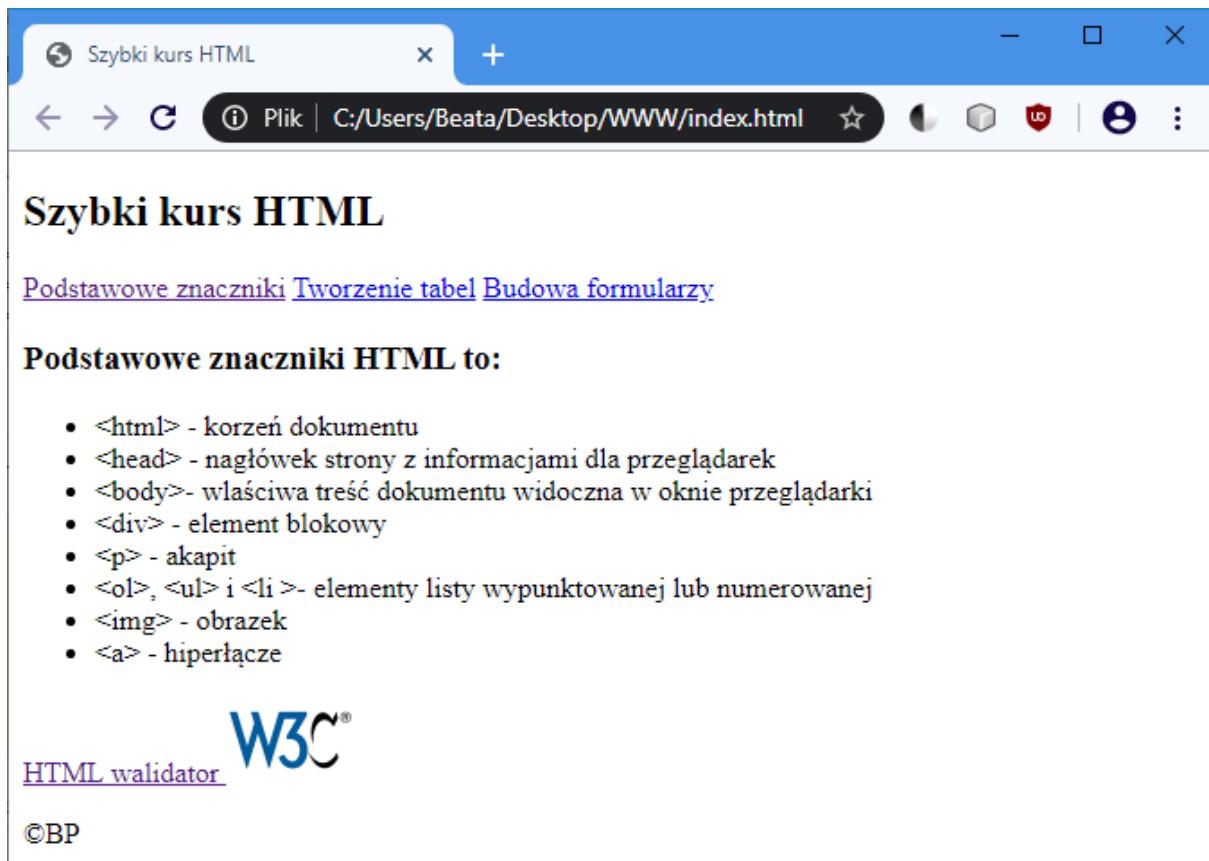
Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny



Rys.1.2. Przykładowy wygląd strony

### UWAGA

Przed publikacją serwisu niezbędna jest walidacja tworzonego kodu. Wykorzystaj dostępny online validator organizacji W3C, która zajmuje się standardami sieciowymi. Validator jest dostępny na stronie: <http://validator.w3.org/>

### Zadanie 1.3. Walidacja kodu dokumentu HTML

Sprawdź czy tworzony dokument *index.html* nie zawiera błędów walidacji. Wykorzystaj w tym celu link z dokumentu do strony validatora i uaktywnij zakładkę **Validate by File Upload**. Wskaż plik do walidacji. Jeśli uzyskasz efekt widoczny na rys.1.3 to możesz kontynuować pracę i realizować kolejne zadania. Jeśli nie udało się – dostaniesz listę błędów ze wskazówkami jak je należy poprawić.



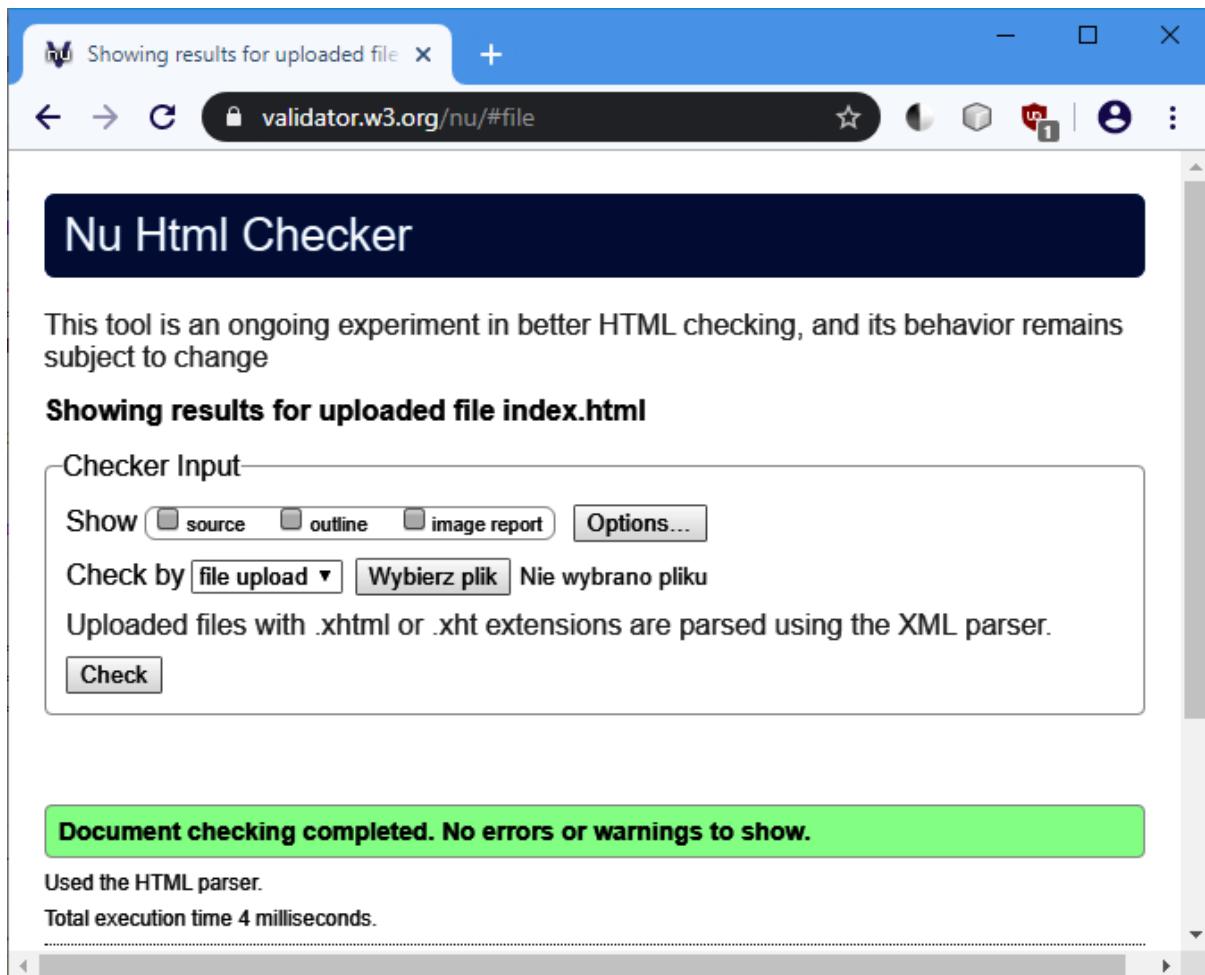
Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny



Rys. 1.3. Efekt pomyślnej walidacji

#### Zadanie 1.4. Tworzenie tabel

- Utwórz pusty dokument i skopiuj do niego całą zawartość pliku *index.html*. Usuń tylko zawartość znacznika *div* o *id="tresc"*, a pozostałe elementy pozostaw bez zmian. Tak zmodyfikowany dokument zapisz jako *tabele.html* i *formularze.html* w tym samym folderze co plik *index.html* (w przykładzie pracujemy z folderem *WWW*). W obu dokumentach odpowiednio zmodyfikuj zawartość znacznika *title*. Ponownie sprawdź działanie linków w dokumencie *index.html*. Ponieważ mamy już właściwie nazwane pliki – linki z menu powinny działać bez problemów. Pamiętaj o ustawieniu odpowiedniej strony kodowej, żeby nie było problemów np. z polskimi literkami.
- Wypełnij treścią z tabelką zawartość elementu o *id="tresc"*, tak jak pokazano na rys. 1.5. Poniżej listy znaczników wstaw przykładową tabelę. Skorzystaj z listingu 1.4, który zawiera prosty schemat tabeli (Rysunek 1.5). Zauważ, że niektóre komórki tabeli z rys.1.5 zostały scalone (w tym celu należy odpowiednio wykorzystać atrybuty *colspan* lub *rowspan* stosowane tylko dla znacznika *td* lub *th*).
- Sprawdź poprawność dokumentu *tabele.html* za pomocą validatora.



Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny

**Listing 1.4. Prosty szablon tabeli**

```
<table border="1">  
  <caption>Wyniki sesji</caption>  
  <thead>  
    <tr>  
      <th>Przedmiot</th>  
      <th>Ocena końcowa</th>  
    </tr>  
  </thead>  
  <tbody>  
    <tr>  
      <td>Matematyka</td>  
      <td>4.0</td>  
    </tr>  
    <tr>  
      <td>Elektrotechnika</td>  
      <td>3.0</td>  
    </tr>  
    <tr>  
      <td>Informatyka</td>  
      <td>5.0</td>  
    </tr>  
  </tbody>  
  <tfoot>  
    <tr>  
      <td>Średnia</td>  
      <td>4.0</td>  
    </tr>  
  </tfoot>  
</table>
```

Wyniki sesji	
Przedmiot	Ocena końcowa
Matematyka	4.0
Elektrotechnika	3.0
Informatyka	5.0
Średnia	4.0

Rys.1.4. Widok tabeli z listingu 1.4 w przeglądarce

W celu scalenia komórek tabeli, dla elementu **td** lub **th** stosuje się atrybuty:

- **colspan** - w celu połączenia kilku komórek leżących obok siebie w kolumnie, np.  
`<td colspan = "3" >`
- **rowspan** – w celu połączenia komórek leżących obok siebie w wierszu np.  
`<td rowspan = "2" >`



Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny

The screenshot shows a web browser window titled "Tworzenie tabel". The address bar displays "Plik | C:/Users/Beata/Desktop/WWW/tabele.html". The main content area is titled "Szybki kurs HTML" and contains the following text:

Podstawowe znaczniki Tworzenie tabel Budowa formularzy

**Podstawowe znaczniki do tworzenia tabel to:**

- <table> - znacznik zawierający całą tabelę
- <caption> - tytuł tabeli
- <thead> - nagłówek tabeli
- <tbody> - zawartość tabeli
- <tfoot> - stopka tabeli
- <tr> - wiersz tabeli
- <td> - komórka tabeli (zawarta w wierszu)
- <th> - komórka nagłówkowa tabeli (również zawarta w wierszu)

**Przykładowa tabela HTML:**

Język programowania	Pozycja	
	Luty 2020	2015
Java	1	2
C	2	1
Python	3	7
C++	4	4
C#	5	7

Na podstawie:  
<http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>

©BP

Rys.1.5. Widok dokumentu tabela.html w przeglądarce



Fundusze Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska

Unia Europejska  
Europejski Fundusz Społeczny



## LABORATORIUM 1. CZĘŚĆ 2. TWORZENIE GRAFICZNEGO INTERFEJSU UŻYTKOWNIKA NA BAZIE PODSTAWOWYCH ELEMENTÓW FORMULARZY HTML.

### Cel laboratorium:

Celem zajęć jest przygotowanie prostego interfejsu użytkownika aplikacji internetowej z wykorzystaniem podstawowych pól formularzy HTML.

### Zakres tematyczny zajęć:

Budowa projektu aplikacji HTML5 w środowisku NetBeans IDE (lub innym wybranym przez studenta).

Budowa formularza z wykorzystaniem pól tekstowych, list rozwijanych, przycisków typu *radio* i *checkbox*.

Walidacja kodu formularza.

Wykorzystanie narzędzi deweloperskich dostępnych w przeglądarce internetowej do obserwacji żądań HTTP.

### Pytania kontrolne:

1. Jakie znaczenie mają atrybuty *action* i *method* w elemencie *<form>*?
2. Jakie znasz podstawowe wartości atrybutu *type* dla pola *<input>*?
3. O czym trzeba pamiętać tworząc grupę przycisków typu *radio*, która umożliwia wybór tylko jednego z nich?
4. Jaki atrybut pola *<select>* umożliwia wybór wielu opcji z listy?
5. Jakie znaczenie w polu *<input>* mają wartości *submit* i *reset* dla atrybutu *type*?

### Zadanie 1.5. Budowa projektu HTML5 w środowisku NetBeans

Dokumenty HTML można tworzyć w dowolnym edytorze tekstowym, ale z uwagi na wygodę warto skorzystać z odpowiedniego środowiska wspomagającego pracę. Jednym z takich środowisk jest projekt *NetBeans* (<https://netbeans.apache.org/download/index.html>), otwarte oprogramowanie mające za zadanie dostarczenie efektywnych narzędzi programowania. NetBeans IDE początkowo było zintegrowanym środowiskiem programistycznym (IDE) dla języka Java, którego głównym celem było (i jest) przyśpieszenie budowy aplikacji Java, w tym również usług sieciowych oraz aplikacji mobilnych. Obecnie wspiera też tworzenie projektów HTML5/JavaScript i PHP.

W celu utworzenia nowego projektu HTML5 w Netbeans:

- a) Wybierz opcję *File->New project*, zaznacz opcje *HTML5/JavaScript* oraz *HTML5/JS Application With Existing Sources* jak na rysunku 1.6.
- b) Następnie w kolejnym oknie wskaż folder roboczy projektu, który będzie traktowany jako główny (*root*) przez serwer WWW, na którym będzie testowana aplikacja. Można zmienić domyślną nazwę projektu (Rys. 1.7).



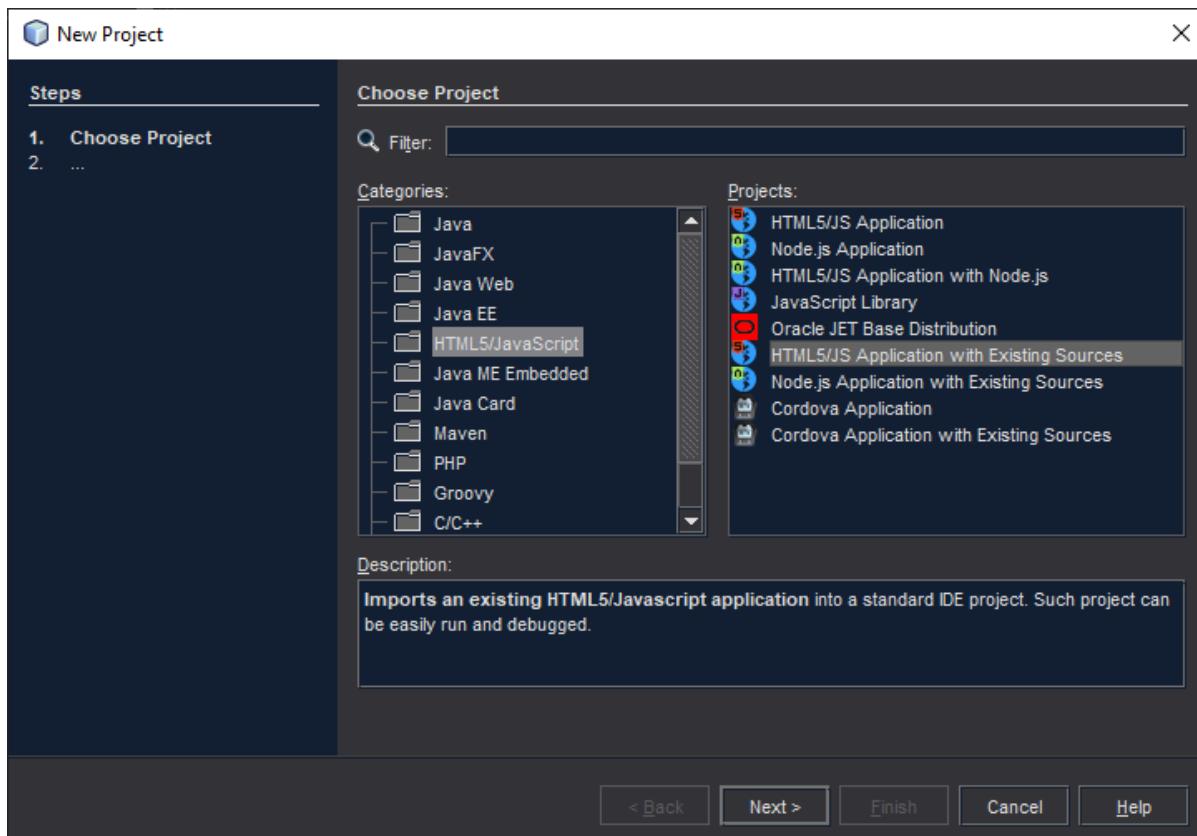
Fundusze Europejskie  
Wiedza Edukacja Rozwój



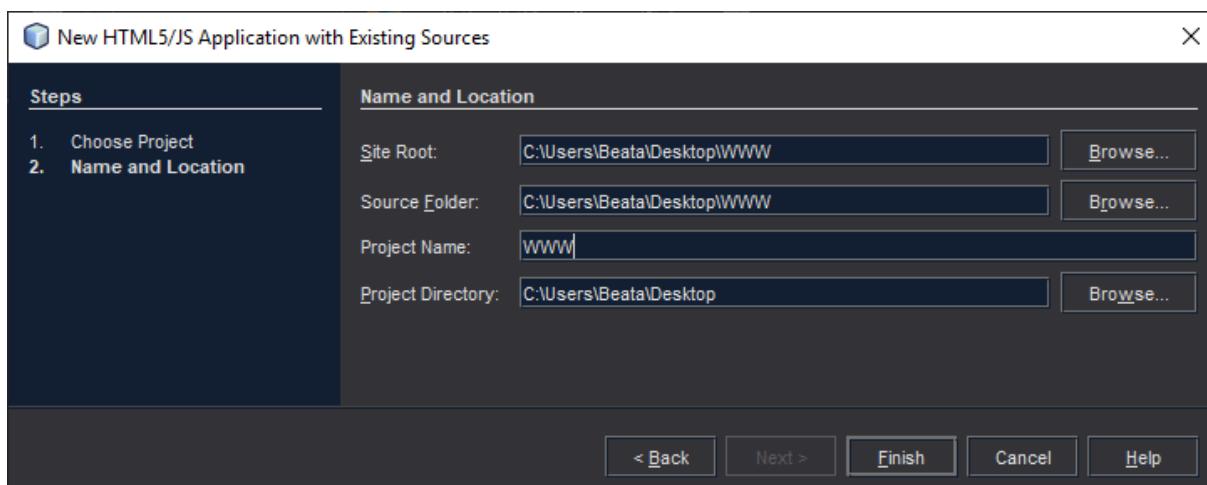
Rzeczpospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny

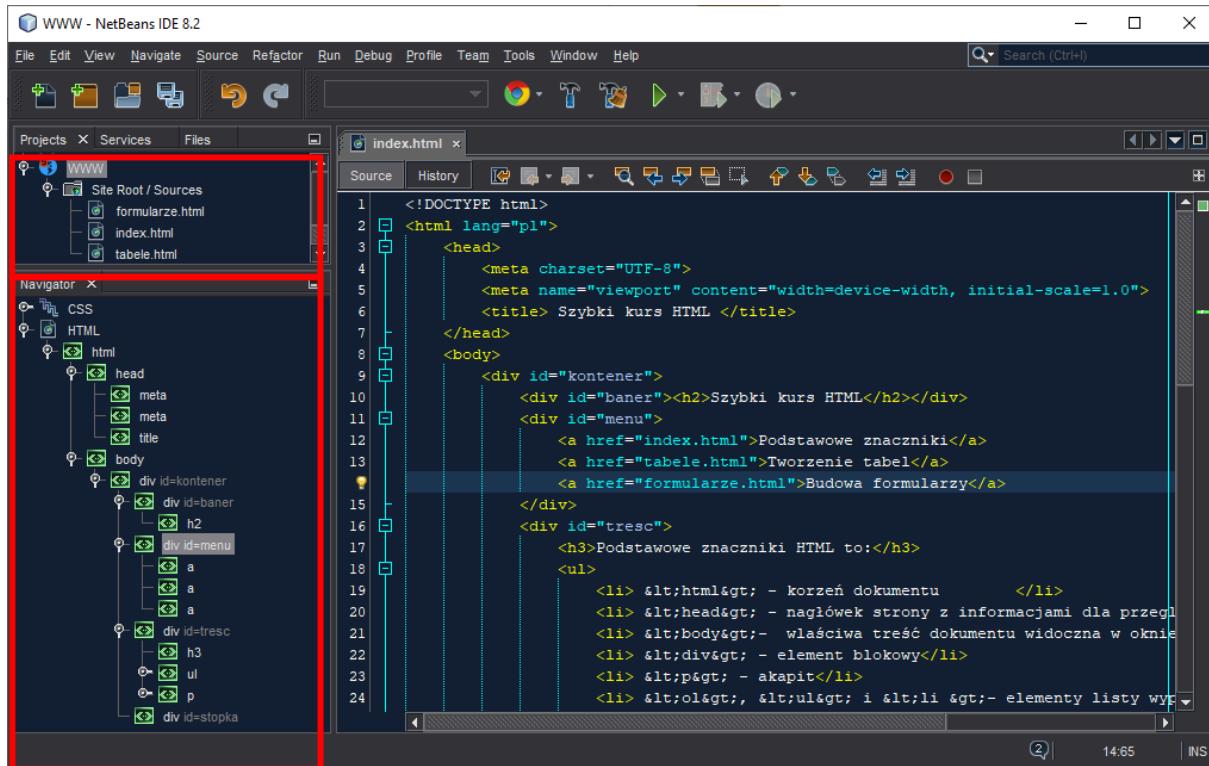


Rys.1.6. Tworzenie projektu HTML5 w NetBeans na podstawie istniejących już źródeł



Rys.1.7. Wskazanie lokalizacji i nadanie nazwy dla projektu

c) Po wykonaniu punktu a) i b) powinieneś uzyskać efekt podobny jak na rysunku 1.8.



Rys.1.8. Zasoby naszego projektu WWW widoczne w NetBeans

- d) Uruchom projekt w NetBeans (ikona zielonej strzałki lub opcja *Run->Run Project*). W wyniku otrzymasz widok w przeglądarce, która jest ustawiona jako domyślna dla NetBeans. Zwróć uwagę, że teraz strona jest już wdrożona i wyświetlna na lokalnym serwerze WWW (*localhost* na porcie *8383*) jest udostępniany i uruchamiany przez NetBeans (porównaj adresy widoczne na rysunku 1.2 i 1.9). Plik *index.html* jest traktowany i uruchamiany domyślnie jako strona początkowa projektu. Jeśli chcesz zobaczyć w przeglądarce (z poziomu serwera) efekt działania innego dokumentu niż *index.html* – można uruchomić aktualnie edytowany plik za pomocą opcji: *Run->Run File*.

The screenshot shows a web browser window with two tabs: 'Tworzenie tabel' and 'Szybki kurs HTML'. The address bar shows the URL 'localhost:8383/Desktop/index.html', which is highlighted with a red box. The main content area of the browser displays the heading 'Szybki kurs HTML' and a section titled 'Podstawowe znaczniki' with three links: 'Tworzenie tabel', 'Tworzenie tabel', and 'Budowa formularzy'. Below this, there is a list of HTML tags with their descriptions.

Tag	Opis
<html>	- korzeń dokumentu
<head>	- nagłówek strony z informacjami dla przeglądarek
<body>	- właściwa treść dokumentu widoczna w oknie
<div>	- element blokowy
<li>	- akapit
<ul>	- element listy

Rys.1.9. Wynik po uruchomieniu projektu na localhost za pomocą NetBeans

### Zadanie 1.6. Budowa formularza

- a) Uzupełnij plik *formularz.html* zawartością jak pokazano na rys.1.11. Skorzystaj z przykładu na listingu 1.5, którego efekt działania przedstawia rysunek 1.10. Zwróć uwagę na postać komentarza w dokumencie HTML oraz zauważ, że **każde pole formularza** (poza przyciskami z jednej grupy radio) posiada unikatowy atrybut **name**. Przy przesyłaniu danych z formularza na serwer, wartość atrybutu **name** jest kluczem, który po stronie serwera pozwala odebrać parametry przekazane wraz z żądaniem HTTP. Wartość wpisana w polu formularza jest przesyłana w żądaniu HTTP w postaci pary klucz=wartość (**key=value**), gdzie kluczem jest wartość atrybutu **name** pola formularza, a wartością klucza jest wartość atrybutu **value** tego pola). Można to podejrzeć za pomocą odpowiednich narzędzi udostępnianych przez przeglądarki (o czym będzie dokładniej w następnych zadaniach). Każde pole formularza może również posiadać atrybut **id**, który z kolei jest bardzo przydatny zarówno do formatowania wyglądu formularza za pomocą stylów CSS, jak i dla skryptów JavaScript, które umożliwiają pracę po stronie klienta z danymi wprowadzonymi do pól formularza.
- b) Sprawdź poprawność dokumentu *formularze.html* za pomocą validatora.

#### Listing 1.5. Budowa przykładowego formularza

```
<div>
<form action="mailto:beatap@cs.pollub.pl" method="post">
    <!-- Przykładowe typy znacznika input -->
    Podaj login : <input type="text" name="login"/> <br />
    Podaj hasło: <input type="password" name="pass"/> <br /><br />
    Wybierz ulubione języki programowania: <br />
    <input type="checkbox" name="C" value="C" /> Język C/C++
    <input type="checkbox" name="Java" value="J" /> Język Java
    <input type="checkbox" name="PHP" value="PHP" /> Język PHP <br />
    Wybierz przeglądarkę, z której korzystasz najczęściej: <br />
    <input type="radio" name="przegladarka" value="Ff" /> Firefox <br />
    <input type="radio" name="przegladarka" value="Ch" /> Chrome <br />
    <input type="radio" name="przegladarka" value="II" />
        Internet Explorer<br />
    <input type="submit" value="Wyślij" />
    <input type="reset" value="Wyczysć" />

    <!-- Inne znaczniki pól formularza -->
    <br />Wskaż hobby i uzasadnij dlaczego to lubisz:
    <br />
    <select size="3" name="hobby" multiple="multiple">
        <option value="t">Turystyka</option>
        <option value="m">Muzyka</option>
        <option value="f">Film</option>
        <option value="s">Sport</option>
        <option value="i">Inne</option>
    </select><br />
    <textarea cols="10" rows="3" name="opinia">
        Po prostu lubię.
    </textarea><br />
    <button name="Kliknij" value="Klikniety">
        Kliknij tutaj
    </button>
</form>
</div>
```



Podaj login :

Podaj hasło:

Wybierz ulubione języki programowania:

Język C/C++  Język Java  Język PHP

Wybierz przeglądarkę, z której korzystasz najczęściej:

Firefox  
 Chrome  
 Internet Explorer

Wskaż hobby i uzasadnij dlaczego to lubisz:

Film  
Sport  
Inne

Po prostu lubię.

Rys. 1.10. Formularz z listingu 1.5.

Szybki kurs HTML

Podstawowe znaczniki Tworzenie tabel Budowa formularzy.

**Podstawowe znaczniki formularza HTML to:**

- <form> - znacznik podstawowy zawierający pola formularza
- <input> - znacznik umożliwia wstawienie różnych pól (pole tekstowe, przycisk typu radio, przycisk typu checkbox) w zależności od wartości jego atrybutu type
- <select> - pole formularza typu lista rozwijana
- <textarea> - pole formularza typu obszar tekstowy
- <button> - pole formularza typu przycisk

**Przykładowy formularz HTML:**

Nazwisko:

Wiek:

Państwo:

Adres e-mail:

**Zamawiam tutorial z języka:**

PHP  C/C++  Java

**Sposób zapłaty:**

eurocard  visa  przelew bankowy

GBP

Rys. 1.11. Gotowy dokument formularze.html w przeglądarce



Fundusze Europejskie  
Wiedza Edukacja Rozwój



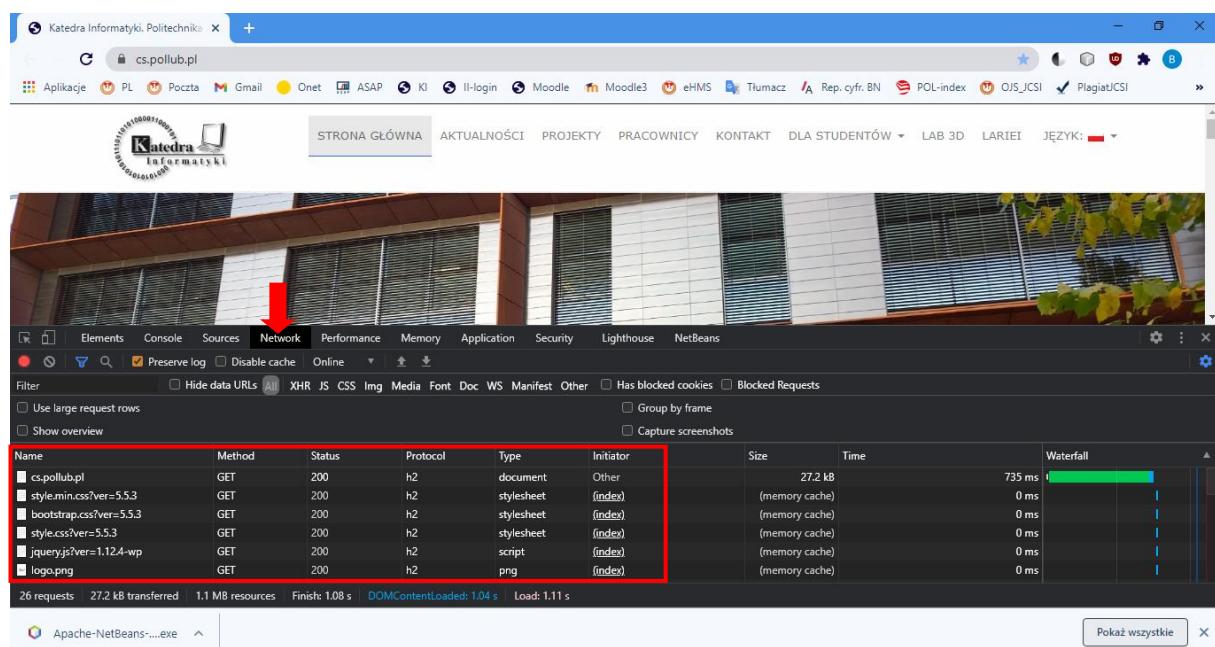
Rzeczypospolita  
Polska

Unia Europejska  
Europejski Fundusz Społeczny



### Zadanie 1.7. Narzędzia deweloperskie w przeglądarce – żądania HTTP

- Sprawdź, ile **żądań HTTP** (i o jakie zasoby?) wysyła przeglądarka w odpowiedzi na nasze **jedno** żądanie o stronę np. <https://cs.pollub.pl/>. W tym celu uruchom narzędzia deweloperskie dostępne w przeglądarce (najczęściej za pomocą klawisza funkcyjnego F12) i wskaż zakładkę **Network** (w Chrome) lub **Sieć** (w Firefox), a następnie odśwież zawartość strony i przejrzyj listę wysłanych żądań http/1.1 lub HTTP/2 (Rys. 1.12).
- Sprawdź postać **żądań HTTP** na przykładzie naszej strony z formularzem z rys. 1.12 – uruchom narzędzie za pomocą **F12** i wskaż zakładkę **Network** (w Chrome) lub **Sieć** (w Firefox), a następnie odśwież zawartość strony z formularzem (Rys. 1.13).
- Uzupełnij dane w formularzu, a następnie spróbuj go wysłać. Na liście żądań (Rys. 1.14) pojawi się kolejne żądanie **mailto:....** do przesłania metodą **POST**. Po kliknięciu na to żądanie sprawdź wartości nagłówków (**Request headers**) oraz listę parametrów (**Query string**) dołączonych do żądania (pobranych z pól formularza).
- Przy opcji **Query string parameters** sprawdź działanie opcji **view source/view parsed** oraz **view URL encoded/view decoded** (zapoznaj się z uwagą na kolejnej stronie).



Rys. 1.12. Lista żądań HTTP dla strony Katedry Informatyki (w Chrome)



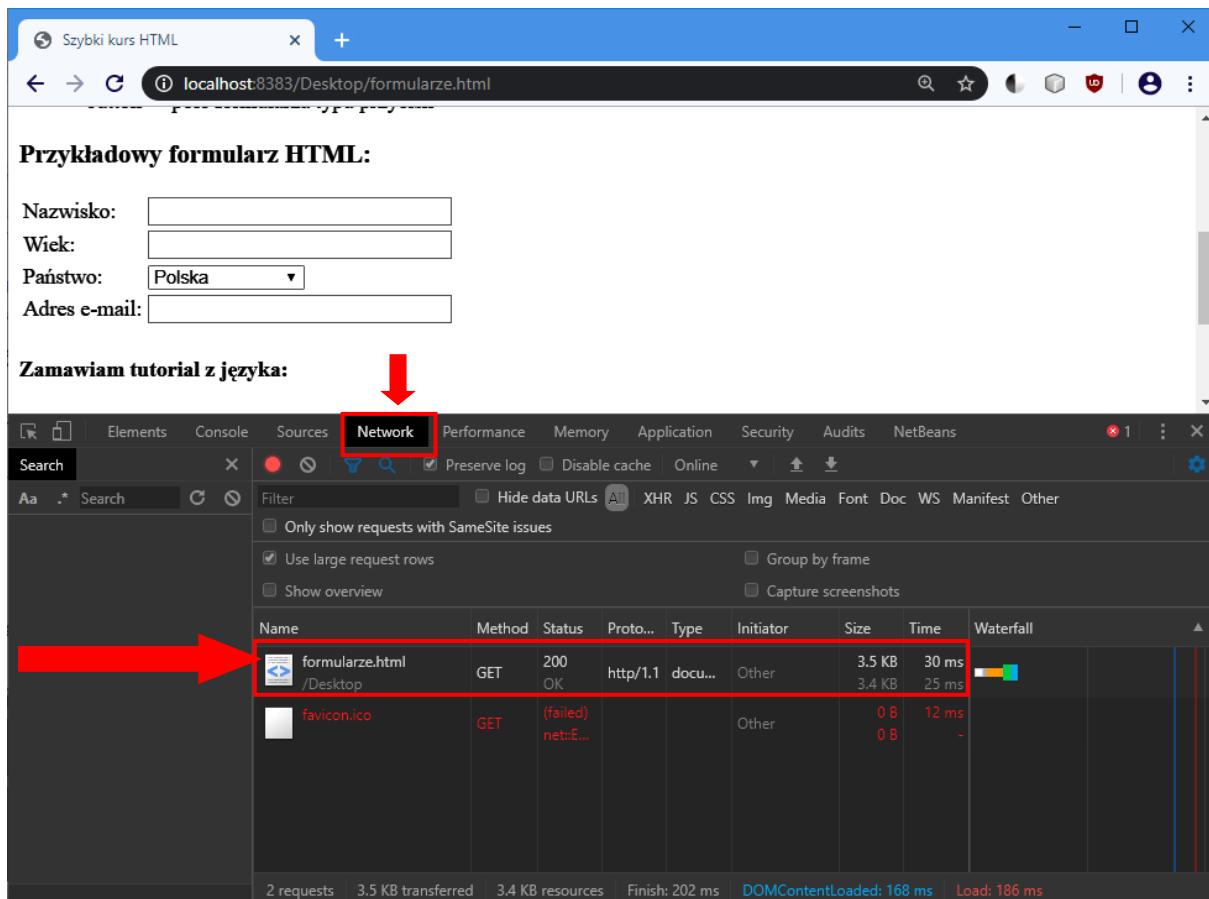
Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska

Unia Europejska  
Europejski Fundusz Społeczny





Rys.1.13. Lista żądań HTTP dla strony z formularzem (Chrome)

### UWAGA

**Kodowanie procentowe** (ang. percent-encoding) lub **kodowanie URL** (ang. URL encoding) to mechanizm kodowania informacji w URI, używany głównie do kodowania danych przesyłanych przez zapytanie GET w adresie URL lub danych z formularza. Dany bajt po zakodowaniu zaczyna się znakiem %, a kodowanie bajtu polega na jego zamianie na dwucyfrową wartość heksadecymalną zapisaną w kodzie ASCII (każdy zakodowany bajt jest zapisany za pomocą trójki znaków). Znaki A-Z, a-z, 0-9, oraz '-', '\_', '.', '~' nie są kodowane. Spacje mogą być zamieniane na znak '+'. Jeśli znak ma specjalne znaczenie w URI (np. '/') to nie jest kodowany, chyba że występuje w danych (wtedy podlega on kodowaniu do '%2F'). Podobnie jest ze znakami specjalnymi '?', '=', '&'. Litery spoza zakresu ASCII zostają zakodowane (np. litera 'Ł' jest kodowana jako '%C5%81', ponieważ w UTF8 ta litera jest zapisywana za pomocą dwóch bajtów, i obydwa trzeba zakodować procentowo).



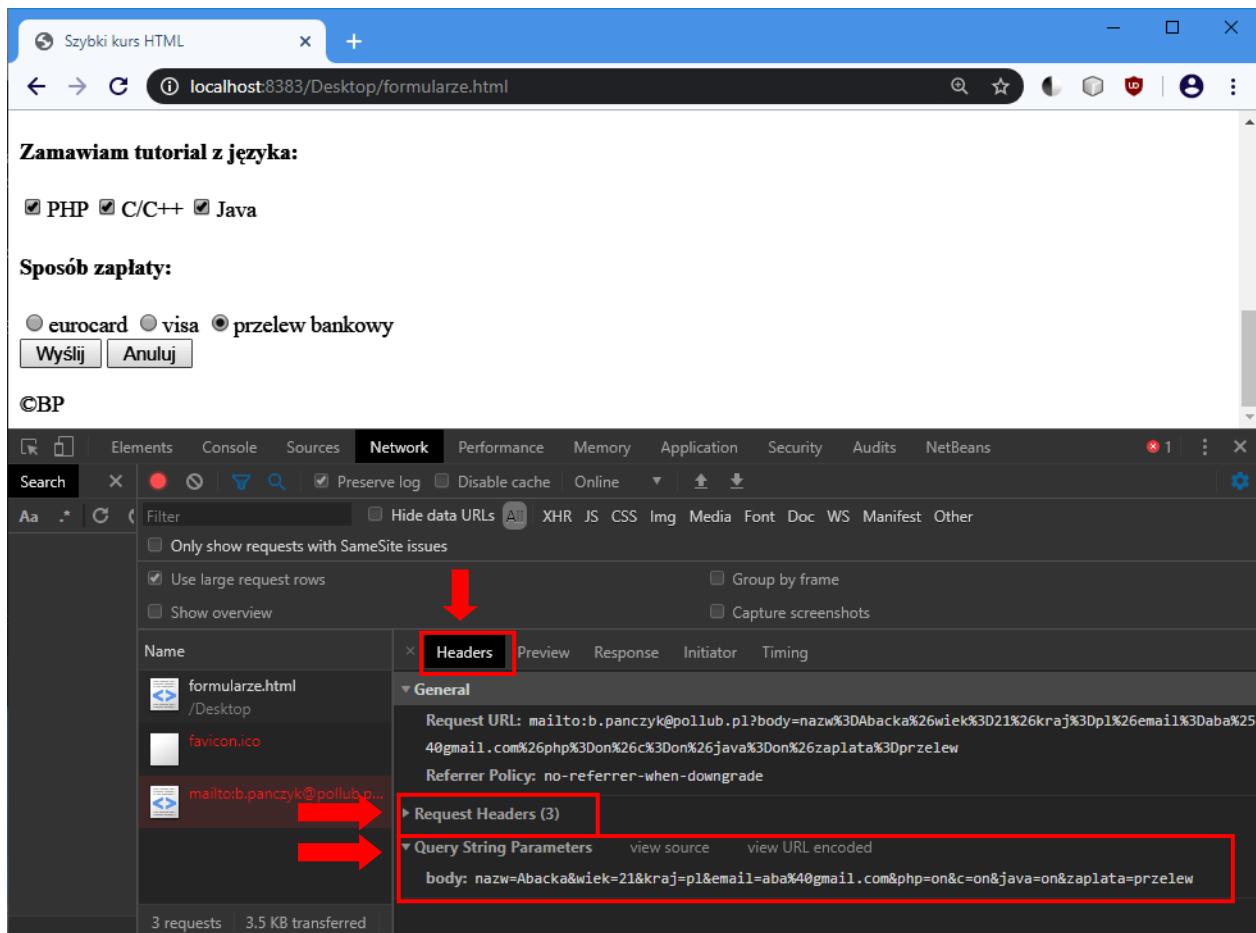
Fundusze Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny



Rys. 1.14. Nagłówki i parametry żądań HTTP

### Zadanie 1.8. Formularz zamówienia pizzy

Do projektu dodaj nowy dokument **zamowienie.html**, w którym stwórz formularz kreatora pizzy (Rys. 1.15) z następującymi wymaganiami:

1. Dane z formularza powinny być przesyłane mailem metodą POST.
2. Poszczególne sekcje formularza (**Rozmiar**, **Dodatki**, **Sosy**) powinny być wydzielone za pomocą znacznika **<fieldset>**.
3. Lista rozwijana **<select>** wyboru sosu ma umożliwiać zamówienie wielu elementów jednocześnie.
4. Do wprowadzania innych uwag – zastosuj element **<textarea>** (o rozmiarze 5 wierszy i 70 kolumn).
5. Formularz powinien zawierać przycisk wysyłania i resetowania danych.
6. Po przygotowaniu formularza, podejmij próbę jego wysłania i podejrzyj łańcuch parametrów (**Query String**) utworzony na podstawie danych wprowadzonych (lub zaznaczonych) w formularzu. Zwróć uwagę na to, w jaki sposób są dołączone dane z pól typu **radio** i **checkbox** oraz z pola **<select>** do wyboru **wielu** sosów.



Fundusze Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita Polska



Unia Europejska  
Europejski Fundusz Społeczny

The screenshot shows a web browser window with the title 'Kreator pizzy'. The URL in the address bar is 'localhost:8383/NI\_2020/pizza...'. The page content is as follows:

- Rozmiar:** A group of three radio buttons:  Mała,  Średnia,  Duża.
- Dodatki:** A list of checkboxes:
  - Szynka
  - Salami
  - Boczek
  - Papryka
  - Pomidor
  - Ser
- Sosy:** A dropdown menu labeled 'Wybierz sos:' with options:
  - Pomidorowy
  - Czosnkowy
  - Ketchup łagodny
  - Ketchup pikantny
- Inne uwagi:** A text input field.
- Buttons:** Two buttons at the bottom: 'Wyczyść dane' and 'Wyślij zamówienie'.

Rys.1.15. Formularz kreatora pizzy w przeglądarce



Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska

Unia Europejska  
Europejski Fundusz Społeczny



## LABORATORIUM 2. WYKORZYSTANIE PODSTAWOWYCH ARKUSZY CSS DO FORMATOWANIA STRONY WIZUALNEJ DOKUMENTU HTML. WALIDACJA KODU DOKUMENTÓW HTML I CSS.

### Cel laboratorium:

Celem zajęć jest zapoznanie się z podstawowymi zasadami formatowania wyglądu stron WWW za pomocą kaskadowych arkuszy stylów CSS.

### Zakres tematyczny zajęć:

Rozbudowanie projektu tworzonego na poprzednich zajęciach o arkusze CSS.

Wybór elementów do formatowania za pomocą podstawowych deskryptorów modelu Box Model.

Przygotowanie arkuszy z opisem reguł do formatowania dokumentów HTML.

Walidacja kodu CSS.

Wykorzystanie narzędzi deweloperskich dostępnych w przeglądarce internetowej do kontrolowania wartości wybranych elementów dokumentów HTML.

### Pytania kontrolne:

1. W jaki sposób można dołączać reguły CSS do dokumentu HTML?
2. Podaj przykłady selektorów i deskryptorów stosowanych w arkuszu CSS.
3. Co to jest Box Model?
4. Wyjaśnij pojęcie klasy i wyjątku w CSS? Jakie atrybuty odpowiadają im w elemencie HTML?

### Wprowadzenie

Do zadań wykorzystaj projekt i pliki z laboratorium 1, na którym zajmowaliśmy się tylko treścią serwisu. Za wizualną stronę serwisu odpowiedzialne są arkusze CSS. Za pomocą odpowiednich reguł możemy dowolnie zdefiniować wygląd każdego znacznika HTML. Najlepszym sposobem jest zdefiniowanie reguł CSS w oddzielnym pliku i dołączenie go znacznikiem *<link>* do wybranego dokumentu HTML.

W folderze projektu (w przykładzie jest to folder *WWW*) utwórz dodatkowo katalog *CSS*, w którym znajdą się pliki z regułami formatowania wyglądu stron. Pliki te są plikami tekstowymi, ale muszą posiadać rozszerzenie *.css*. Struktura naszego projektu z przykładowym plikiem CSS powinna być następująca:

- *WWW – index.html*
- *tabele.html*
- *formularze.html*
- *CSS – style.css*

### Zadanie 2.1. Podstawy arkuszy CSS

- a) W NetBeans (lub innym IDE) otwórz swój projekt *WWW* i plik *index.html* oraz w części nagłówkowej dokumentu dodaj do niego nieparzysty (bez zawartości, ale z atrybutami) znacznik *<link>*. Za pomocą tego znacznika można wskazać (atributem *href*), gdzie przeglądarka ma szukać pliku z regułami CSS. Znacznik ten posiada również dwa inne atrybuty: *type* i *rel*. W naszym przykładzie powinno to wyglądać



Fundusze Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny

następującą:

```
<link rel="stylesheet" href="CSS/style.css" type="text/css" />
```

Zauważ, że wskazano względną (wobec głównego folderu projektu WWW) ścieżkę do pliku CSS (można również wskazać tu adres URL do pliku CSS na serwerze zewnętrznym, tak samo jak w przypadku hiperłącza). **Nigdy nie podajemy adresu bezwzględnego** (w lokalnym systemie plików).

- b) Do projektu dodaj nowy folder CSS (*File/New File*, wybierz *Other i Folder*) oraz plik *style.css* umieszczony w tym folderze (*File/New File*, oraz *Cascading Style Sheet*). Do stworzonego dokumentu *style.css* wpisz pierwszą regułę formatującą tło całego dokumentu i koloru tekstu (skorzystamy w tym celu ze znacznika *<body>*):

```
body {  
    background: #4ae;  
    color:#000040;  
}
```

Do wyboru koloru możesz posłużyć się pomocniczą mapką kolorów (w NetBeans mapkę można uzyskać po wcisnięciu klawiszy *Ctrl+Space* i wskazaniu pierwszej opcji: *Color chooser* w momencie kiedy po znaku dwukropka „:” w regule CSS spodziewany jest kod koloru).

Podana powyżej reguła CSS opisuje tylko kolor tła i kolor tekstu znacznika *body*. Wszystkie elementy zawarte w *body* będą **dzierdzicyły** te ustawienia, chyba, że podamy odpowiednie definicje dla kolejnych elementów. Za pomocą odpowiednich parametrów można sformatować: czcionkę, marginesy, obramowanie, wcięcia itp.

- c) Zapisz plik *style.css* i otwórz *index.html* w przeglądarce.  
d) Do pliku *style.css* dodaj kolejną regułę opisującą kolor tła i tekstu dla znacznika *<h1>*, *<h2>* i elementu o *id="stopka"*, ale z odwróconymi kolorami. Dodatkowo zdefiniuj wyrównanie poziome do środka (centrowanie w poziomie):

```
h1, h2, #stopka {  
    background: #000040;  
    color: #8080ff;  
    text-align:center;  
}
```

Dokument w przeglądarce powinien teraz wyglądać jak na rys.2.1.

- e) Do pliku *style.css* dodaj kolejne reguły: ustaw białe tło dla elementu o *id="menu"* i jasnoniebieskie dla elementu o *id="tresc"*. Sprawdź wygląd w przeglądarce (Rys.2.2).  
f) Otwórz pliki *formularze.html* i *tabele.html* i za pomocą znacznika *<link>* również dodaj *style.css* jako arkusz formatujący. Sprawdź efekt w przeglądarce.  
g) Sprawdź poprawność arkusza za pomocą validatora CSS (Rys. 2.3).

## UWAGA

Przed publikacją serwisu niezbędna jest również walidacja kodu CSS. Odpowiedni validator jest udostępniany online na stronie: <http://jigsaw.w3.org/css-validator/>

Gotowy CSS (uzupełniony dodatkowo wyzerowaniem marginesów i wypełnień dla wybranych elementów) przedstawia listing 2.1. Porównaj go ze swoim arkuszem CSS.



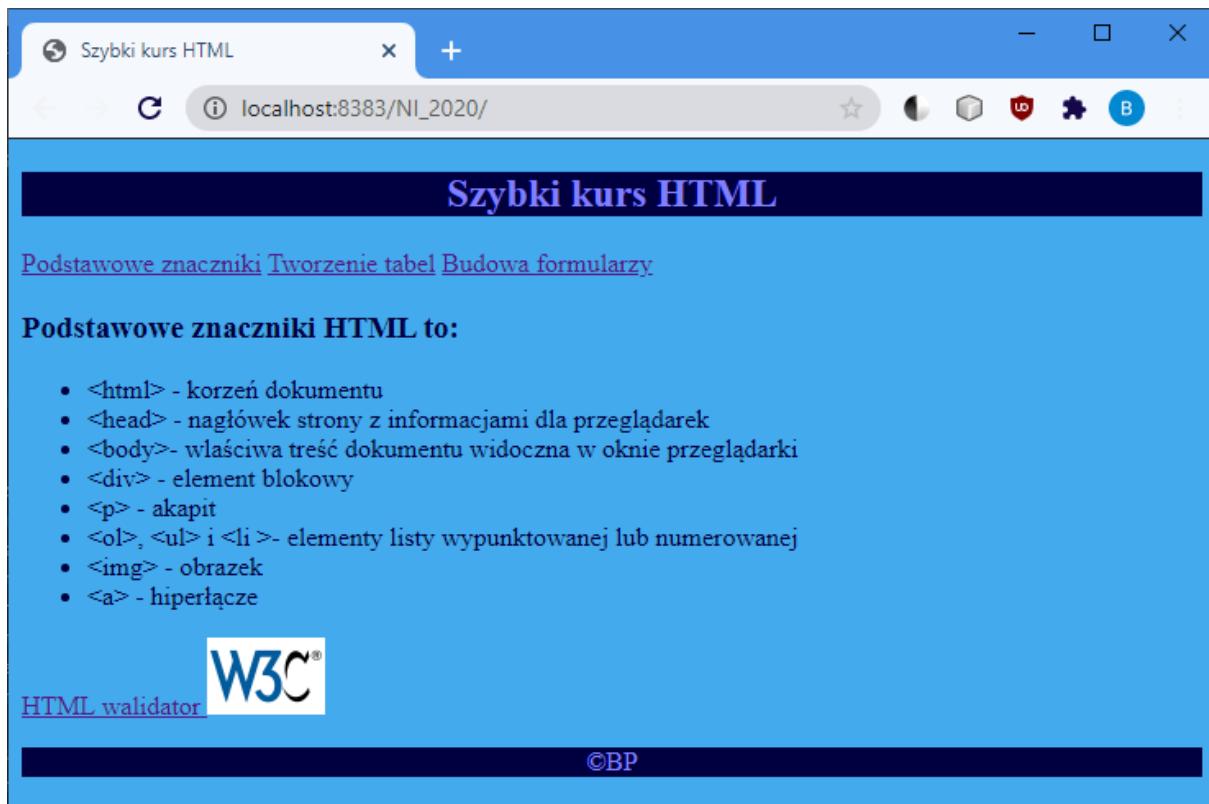
Fundusze Europejskie  
Wiedza Edukacja Rozwój



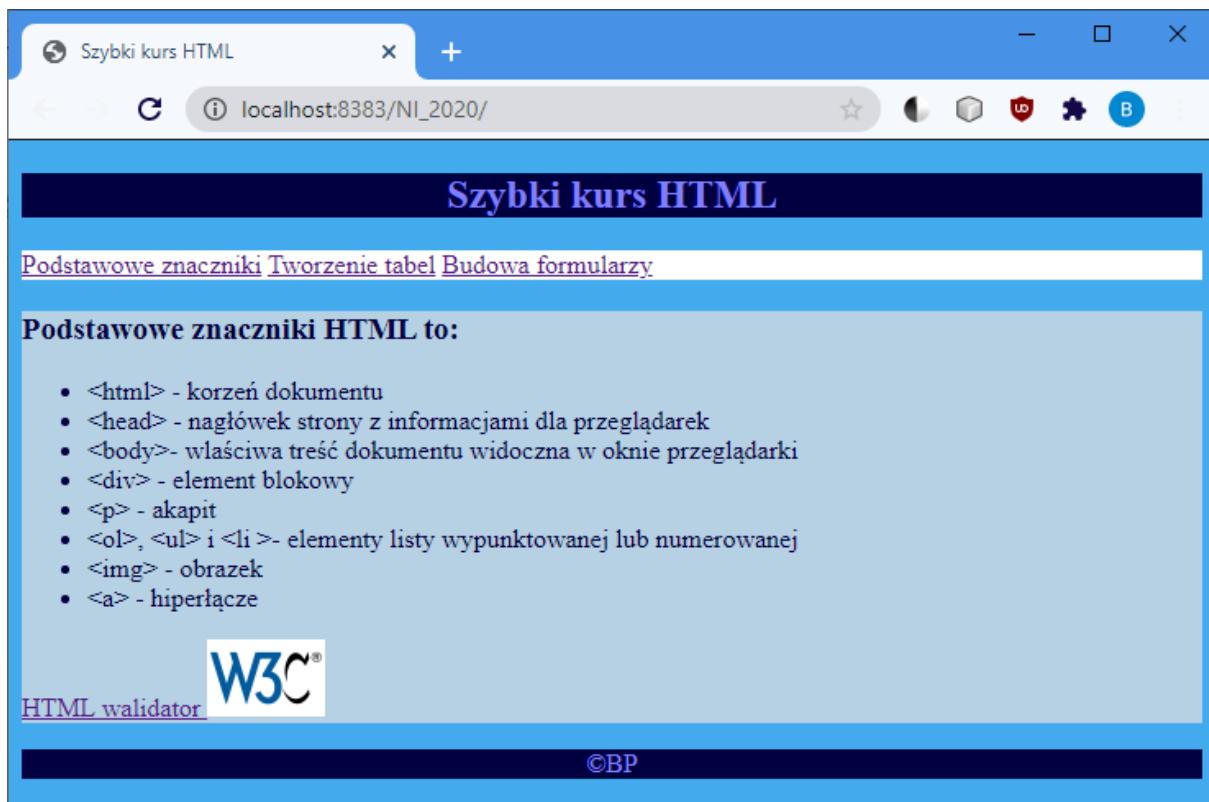
Rzeczpospolita  
Polska



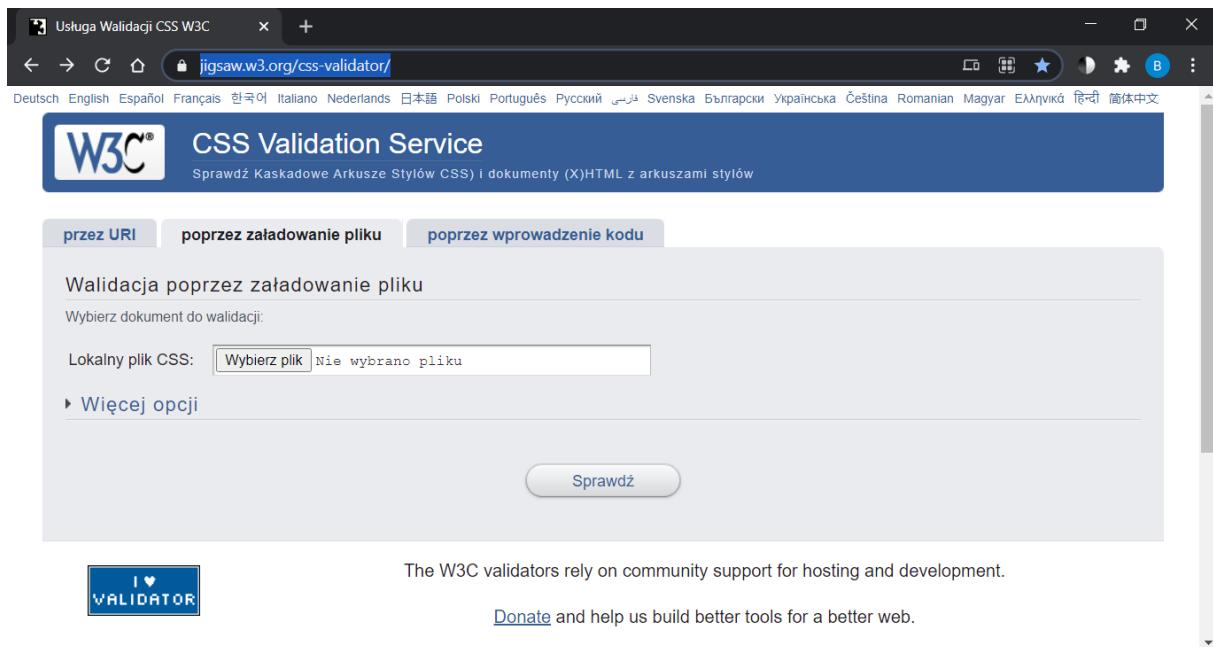
Unia Europejska  
Europejski Fundusz Społeczny



Rys.2.1. Dokument index.html z wykorzystaniem arkusza CSS



Rys.2.2. Dokument index.html z kolejnymi regułami CSS



Rys.2.3. Validator kodu CSS

### Listing 2.1. Plik style.css z komentarzami

```
/* wyzerowanie marginesów i wypełnień: */
h1, h2, h3 {
    margin:0;
    padding:0;
}
/*formatowanie znacznika body:*/
body {
    background:#4ae;
    color:#000040;
}
/*formatowanie znaczników h1, h2 i elementu o identyfikatorze stopka:*/
h1,h2, #stopka {
    background: #000040;
    color: #8080ff;
    text-align:center;
}
/*formatowanie elementu o id="tresc": */
#tresc {
    background:#80ffff;
}

/*formatowanie elementu o id="menu": */
#menu {
    background:#ffffff;
}
```

### Zadanie 2.2. Formatowanie tabeli, klasy elementów

- Utwórz nowy plik *tabele.css* w istniejącym już folderze **CSS**.
- Do pliku *tabele.html* dodaj kolejny znacznik *<link>* ze wskazaniem na plik *tabele.css* oraz dla znacznika *<table>* usuń umieszczony tam wcześniej atrybut *border="1"*



Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



Rzeczypospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny

(teraz obramowanie tabeli sformatujemy w CSS – usuniemy tym samym uwagę validatora HTML5, która informowała o tym aby obramowanie tabel umieszczone było w pliku z regułami CSS, a nie w kodzie dokumentu HTML poprzez atrybut *border* dla elementu *table*).

- c) W pliku *tabele.css* umieść następujące reguły:
- dla całej tabeli (**table**): tło jasnożółte, tekst wyśrodkowany, szerokość tabeli 94% (względem elementu, w którym jest zawarta tabela), margines (**margin**) – 2%, obramowanie (**border**): 1pt, granatowe, wybrany styl (np. *solid*);
  - tytuł tabeli (**caption**): czcionka 120%, pogrubiony tekst;
  - nagłówek i stopka tabeli (**thead**, **tfoot**): ciemnogranatowe tło, tekst biały, wielkość czcionki 110% czcionki bazowej, wypełnienie (**padding**) 5pt’;
  - każda komórka (**td**, **th**): obramowanie (**border**) 1pt, linia ciągła, granatowa;
  - hiperłącza *<a>* zagnieździone w *<tfoot>* ustal kolor biały:  

```
tfoot a {color:#ffffff;}
```

Sprawdź efekt w przeglądarce (Rys. 2.4).
- d) Uzupełnij plik *tabele.css* tak, aby parzyste wiersze tabeli miały tło białe – w tym celu skorzystamy z tzw. **klas elementów**. Definicja takiej klasy w CSS polega na utworzeniu reguły dla wszystkich (lub tylko wybranych) znaczników HTML posiadających atrybut *class="nazwa\_klasy"*. Na przykład reguła definiująca formatowanie klasy o nazwie *parzysty* (dla wszystkich znaczników) w CSS wygląda następująco:
- ```
*.parzysty { background:#ffffff; }
```
- a znacznik w **dokumencie HTML**, który ma skorzystać z klasy *parzysty* musi posiadać atrybut **class** np.
- ```
<tr class="parzysty" > ... </tr>
```
- Sprawdź efekt w przeglądarce (Rys. 2.5).
- e) Sprawdź poprawność dokumentu *tabele.css* za pomocą validatora CSS.

Gotowy arkusz *tabele.css* przedstawia Listing 2. 2.

**Listing 2.2. Przykładowy plik *tabele.css*, uzupełniony komentarzami**

```
/* ustawienia dla całej tabeli: */
table {
    background:#fffffb3;
    text-align:center;
    width:94%;
    margin:2%;
    border:1pt solid #000040;
}

/*ustawienie tytułu tabeli:*/
caption {
    font-size:150%
}

/*formatowanie części nagłówkowej i stopki tabeli: */
tfoot, thead {
    background:#0000a0;
    color:#ffffff;
    font-size:120%
}
```



Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny

```
/*formatowanie pojedynczych komórek zwykłych i nagłówkowych: */
td, th {
    border:1pt solid #000040;
}

/*formatowanie elementu <a> zawartego w elemencie <tfoot>: */
tfoot a {
    color:#ffffff;
}

/* definicja klasy o nazwie parzysty - można
   ją zastosować dla dowolnego znacznika w dokumencie html
   podając atrybut class="parzysty" :*/
*.parzysty {
    background:#ffffff;
}
```

Język programowania	Pozycja	
	Luty 2020	2015
Java	1	2
C	2	1
Python	3	7
C++	4	4
C#	5	6

Na podstawie:  
<http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>

Rys.2.4. Widok tabeli po wykonaniu zad.2.2. a-c.

Język programowania	Pozycja	
	Luty 2020	2015
Java	1	2
C	2	1
Python	3	7
C++	4	4
C#	5	6

Na podstawie:  
<http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>

Rys.2.5. Widok tabeli po wykonaniu całego zad.2.2.

### Zadanie 2.3. Formatowanie formularzy

- Wykorzystując przykłady z wykładów, utwórz kolejny arkusz CSS o nazwie *formularze.css*. Dodaj kolejny znacznik *<link>* do pliku *formularze.html* (lub uzupełnij arkusz *style.css* regułami formatowania dla formularza).
- Do dokumentu z formularzem wprowadź etykiety *<label>* dla wszystkich pól tekstowych.
- Utwórz dowolne reguły CSS ukierunkowane na znaczniki formularzy.
- Sprawdzaj efekty na bieżąco w przeglądarce.

Przeprowadź walidację nowego arkusza.

#### UWAGA

W celu zminimalizowania liczby żądań wysyłanych przez przeglądarkę o kolejne arkusze CSS – można wszystkie reguły CSS umieścić w jednym pliku np. *style.css*.



Fundusze Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny

## **LABORATORIUM 3. POZYCJONOWANIE ELEMENTÓW NA STRONIE INTERNETOWEJ. WYKORZYSTANIE NARZĘDZI DEWELOPERSKICH DOSTĘPNYCH W PRZEGŁĄDARCE INTERNETOWEJ DO KONTROLI ARKUSZY CSS.**

### **Cel laboratorium:**

Celem zajęć jest zapoznanie z możliwościami pozycjonowania elementów strony internetowej za pomocą kaskadowych arkuszy stylów CSS oraz wykorzystanie narzędzi dostępnych w przeglądarce do kontroli i zarządzania regułami CSS.

### **Zakres tematyczny zajęć:**

Rozbudowanie projektu tworzonego na poprzednich zajęciach – pozycjonowanie elementów na stronie internetowej.

Wybór metody do pozycjonowania elementów.

Przygotowanie arkuszy z opisem reguł do tworzenia szablonu strony.

Wykorzystanie narzędzi deweloperskich dostępnych w przeglądarce internetowej do kontrolowania wartości wybranych elementów dokumentów HTML.

### **Pytania kontrolne:**

1. Czym różni się naturalny układ elementów blokowych i liniowych?
2. Na czym polega pozycjonowanie elementów strony HTML?
3. Podaj przykładowe metody stosowane do budowy układów kolumnowych.
4. O czym należy pamiętać określając szerokość elementu za pomocą właściwości ***width***?

### **Wprowadzenie**

Dla przypomnienia struktura naszego projektu po wykonaniu zadań z Lab.1 i 2 jest następująca:

- WWW – *index.html*
  - *tabele.html*
  - *formularze.html*
  - CSS – *style.css*
    - *tabele.css*
    - *formularze.css*

W celu realizacji układu kolumnowego w naszym projekcie zajmiemy się pozycjonowaniem elementów *<div>* o określonych identyfikatorach *id*, układając je na początek w 2 kolumny: element z menu po lewej stronie, element z treścią po prawej. Wykorzystamy pozycjonowanie bazując na tzw. elementach pływających (właściwość *float* w CSS - zobacz materiały wykładowe).

### **Zadanie 3.1. Układ dwukolumnowy**

- a) Otwórz pliki *index.html*, *tabele.html* i *formularze.html* oraz zmodyfikuj element: *<div id="menu">* tak, aby każdy element hiperłącza był elementem listy wypunktowanej *<ul>*:  
*<ul>*



**Fundusze Europejskie**  
Wiedza Edukacja Rozwój



**Rzeczpospolita Polska**



**Unia Europejska**  
Europejski Fundusz Społeczny

```
<li> <a href="index.html">Podstawowe znaczniki</a> </li>
<li> <a href="tabele.html">Tworzenie tabel</a> </li>
<li> <a href="formularze.html">Budowa formularzy</a> </li>
</ul>
```

- [Podstawowe znaczniki](#)
- [Tworzenie tabel](#)
- [Budowa formularzy](#)

b) Sprawdź, czy hiperłącza nadal prowadzą do właściwych stron.

c) Otwórz plik *style.css*, ustaw tło elementu kontenera (*#kontener*) na białe i dodaj reguły pozycjonujące dla elementów *menu*, *tresc* i *stopka* (pamiętaj, aby suma szerokości elementów ustawionych w kolumnie, łącznie z marginesami, ramkami i wypełnieniem nie przekraczała 100%):

```
/* tło kontenera */
#kontener {
    background:#ffffff;
}
/*pozycjonowanie elementów: */
#menu {
    float:left;
    width:20%;
    padding:2%;
}
#tresc {
    float:right;
    width:70%;
    padding:2%;
}
/* umieszczenie stopki poniżej elementów z float */
#stopka{
    clear:both;
    width:100%;
    padding:5px 0 5px 0;
    text-align:center;
}
```

d) Przeprowadź ponownie walidację kodu CSS z pliku *style.css*. Sprawdź efekty w przeglądarce (Rys.3.1).



Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska

Unia Europejska  
Europejski Fundusz Społeczny



Rys. 3.1. Układ dwukolumnowy

### Zadanie 3.2. Układ trójkolumnowy

W celu pokazania logiki układu trójkolumnowego dodamy jeszcze jeden element `<div id="pasek">` (trzecia kolumna), do pliku `index.html`, a następnie zmodyfikujemy pozycjonowanie elementów `menu`, `tresc` i `pasek` w pliku `style.css`.

a) Do pliku `index.html` (po elemencie `tresc`, ale przed elementem `stopka`) dodaj dodatkowy blok `div`. W tym bloku umieść kolejne dwa bloki `div` z atrybutem `class="box"` (pozwoli nam to na późniejsze sformatowanie wyglądu tych bloczków):

```
<div id="tresc">
    <!--Tu zostawiamy to co było -->
</div>
<!--Tu dodajemy nowy blok div o id=pasek:-->
<div id="pasek">
    <!-- W pasku zawarty jest kolejny blok klasy box:-->
    <div class="box">
        <h3>Validatory:</h3>
        <ul>
            <li> <a href="http://validator.w3.org/">HTML </a></li>
            <li> <a href="http://jigsaw.w3.org/css-validator/">CSS</a></li>
        </ul>
    </div>
    <!-- Kolejny blok klasy box:-->
    <div class="box">
        <h3>Kursy:</h3>
        <ul>
            <li> <a href="https://www.w3schools.com/html/"> HTML </a></li>
            <li> <a href="https://www.w3schools.com/css/"> CSS </a></li>
            <li> <a href="https://www.w3schools.com/js/"> JS </a></li>
            <li> <a href="https://www.w3schools.com/php/"> PHP </a></li>
        </ul>
    </div>
</ul>
```



Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska



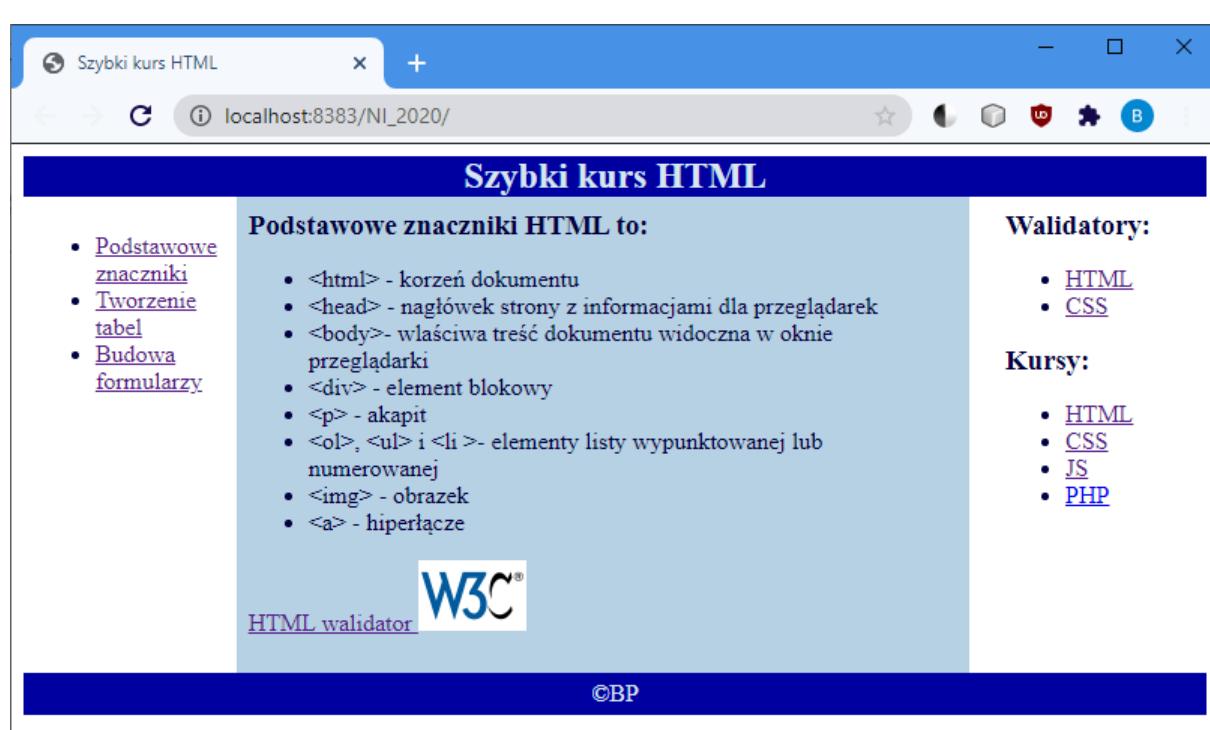
Unia Europejska  
Europejski Fundusz Społeczny

```
</div>
</div><!--Koniec bloku div o id=pasek:--&gt;
<!--Blok stopki pozostaje bez zmian:--&gt;
&lt;div id="stopka"&gt;&amp;copy;BP&lt;/div&gt;</pre>
```

- b) W pliku *style.css* zmodyfikuj część związaną z pozycjonowaniem elementów (wymień poprzednie formatowanie na poniższe):

```
/*pozycjonowanie elementów: */
#menu {
    float:left;      width:16%;
    padding:1%;
}
#tresc {
    float:left;      width:60%;
    padding:1%;
}
#pasek {
    float:right; width:16%;
    padding:1%;
}
#stopka{
    clear:both;
    width:100%;
    padding:5px 0 5px 0;
    text-align:center;
}
```

- c) Efekt końcowy przedstawia rysunek 3.2. Przeprowadź validację kodu HTML i CSS (skorzystaj z linków na pasku po prawej stronie).



Rys. 3.2. Układ trójkolumnowy po modyfikacjach w *index.html* i *style.css*



Fundusze Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska



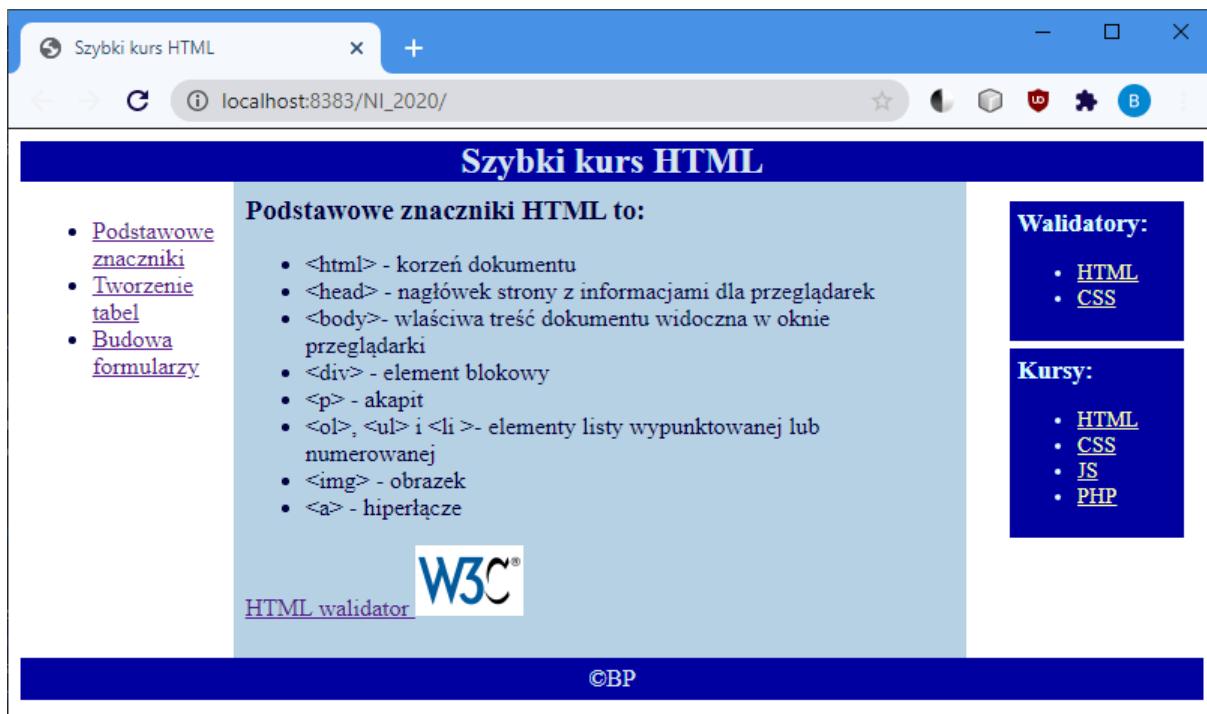
Unia Europejska  
Europejski Fundusz Społeczny

- d) Żeby zachować spójny wygląd całego serwisu, element z paskiem należałoby dodać także do plików *tabele.html* i *formularze.html*. W naszym przykładzie nie jest to konieczne, gdyż kolejne zadania będąmy realizować na układzie dwukolumnowym.

- e) Na koniec do pliku *style.css* dodamy jeszcze formatowanie dla elementu klasy *box*, na przykład (Rys.3.3.):

```
*.box { background:#0000a0;
        color:#dbffff;
        padding:5px;
        margin:5px;
        font-size:90%;}

/* hiperłącza w elementach klasy box*/
.box a{color:#ffffb0}
```



Rys. 3.3. Widok strony ze sformatowanymi elementami box w prawym pasku w układzie trójkolumnowym

### Zadanie 3.3. Formatowanie menu

Ciekawe efekty wizualne dla elementów *menu* uzyskuje się dzięki zastosowaniu odpowiednich reguł CSS.

- a) Zmodyfikuj ponownie układ bloków na dwie kolumny tak, żeby menu znalazło się na górze (Rys. 3.4) i wykorzystaj dodatkowe reguły CSS dla elementów w menu.

Zmodyfikowane fragmenty pliku *style.css* (zwróć uwagę na podobny przykład z materiałów wykładowych) mają postać np.:

```
/*Nawigacja: */
#menu {
    float:left;
    width:100%;
    margin:0;
```



Fundusze Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny

```
padding:0;
background:#ffffff;
border-bottom:2px solid #0000a0;
}
#menu li {
float:left;
margin:0;
padding:0;
list-style:none;
padding-right:3px;
font-family:"Lucida
Grande", sans-serif;
font-size:85%;
}
#menu a {
float:left;
display:block;
margin:0;
padding:4px 8px;
color:#ffffd7;
text-decoration:none;
border: 1px solid
#0000c6;

border-bottom:none;
background:#0000a0;
}
#menu a:hover {
background:#0000d5;
color:#ffffff;
text-decoration:none;
}

/*Pozycjonowanie elementów na stronie */
#tresc {
float:left;
width:75%;
padding:1%;
}
#pasek {
float:right;
width:20%;
padding:1%;
}
#stopka{
clear:both;
width:100%;
padding:5px 0 5px 0;
text-align:center;
}
```



Fundusze  
Europejskie  
Wiedza Edukacja Rozwój

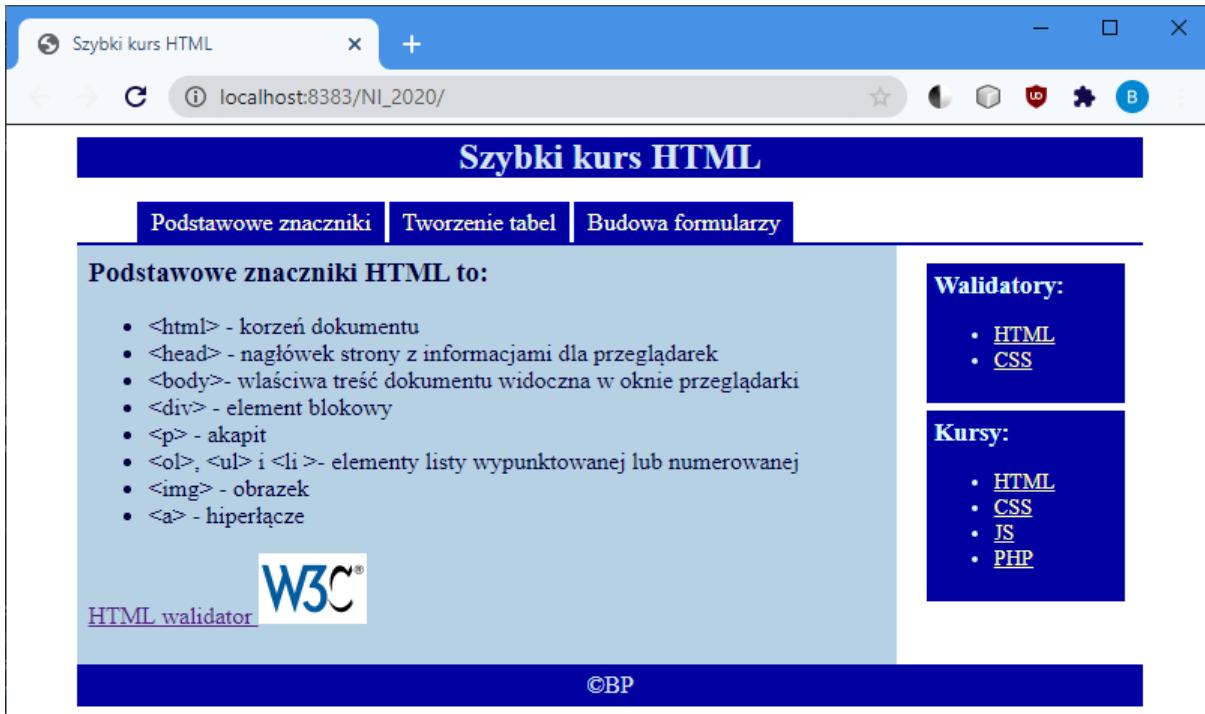


Rzeczpospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny

- b) Ustaw też szerokość elementu kontenera np. na 90% i wycentruj jego położenie na stronie (Rys. 3.4).



Rys.3.4. Sformatowane menu i wycentrowany kontener

#### Zadanie 3.4. Menu rozwijane

Zmodyfikuj teraz zawartość elementu menu naszego serwisu (na wszystkich stronach: *index.html*, *tabele.html* i *formularze.html*) do postaci:

```
<div id="menu">
  <ul id="nav">
    <li><a href="index.html">Podstawy</a></li>
    <li><a href="tabele.html">Tabele</a></li>
    <li><a href="">Formularze</a>
      <ul>
        <li><a href="formularze.html" >Pola formularza</a></li>
        <li><a href="obliczenia.html" >Obliczenia w formularzu</a></li>
      </ul>
    </li>
    <li><a href="" title="">Galeria</a></li>
  </ul>
</div>
```

Zauważ, że rozbudowaliśmy naszą listę nawigacyjną, zagnieżdżając w niej kolejną listę dla elementu **Formularze**. W kolejnym kroku sformatujemy element nawigacyjny tak, aby uzyskać efekt rozwijanego menu (Rys. 3.5).

W pliku *style.css* **wymień** poprzednio utworzone reguły CSS dla nawigacji na następujące:



Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny

```
/*Nowa nawigacja*/
#nav {
    text-align: left;
    overflow: hidden;
    font-family: calibri, tahoma;
}

ul#nav li {
    border: 1px solid #d5d5d5;
    list-style-type: none;
    float: left;
    background: #98bed8;
    margin-left: 3px;
}

ul#nav li:hover {
    border: 1px solid #d5d5d5;
    list-style-type: none;
    float: left;
    background: #f0f0f0;
    margin-left: 3px;
}

ul#nav li a {
    display : block;
    padding : 4pt;
    text-decoration : none;
}

ul#nav li li a {
    width: 160px;
}

ul#nav li li:hover {
    background: white;
}

ul#nav li ul {
    overflow: hidden;
    display: none;
}

ul#nav li:hover ul {
    position: absolute;
    background: #aaa;
    padding: 0;
    display: block;
    width: 177px;
}
```



Fundusze  
Europejskie  
Wiedza Edukacja Rozwój

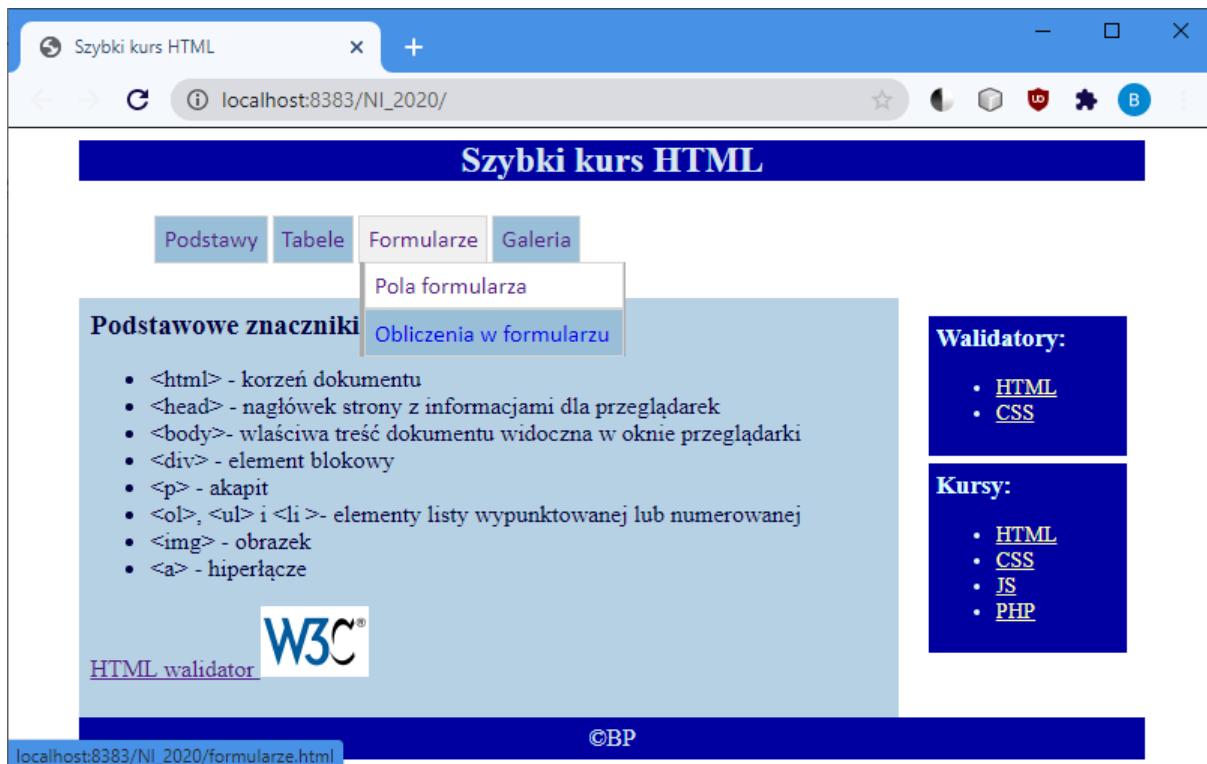


Rzeczpospolita  
Polska

Unia Europejska  
Europejski Fundusz Społeczny



Efekt po wymianie formatowania elementu nawigacyjnego powinien być podobny do tego zaprezentowanego na rysunku 3.5. Dopasuj kolorystykę do własnych preferencji.

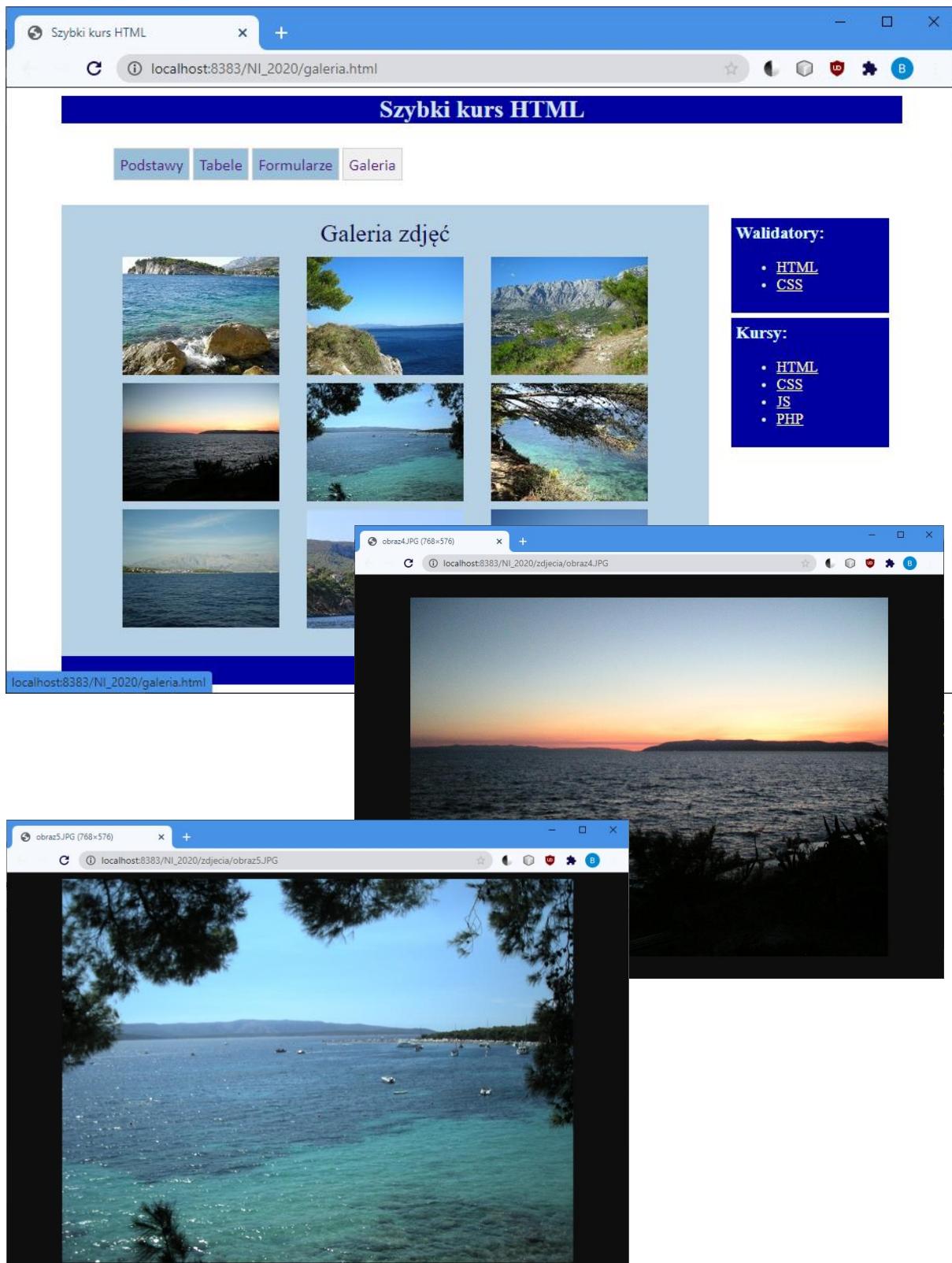


Rys.3.5. Menu po modyfikacjach z rozwijanymi opcjami dla Formularzy i dodanym elementem Galeria

### Zadanie 3.5. Tworzenie galerii zdjęć

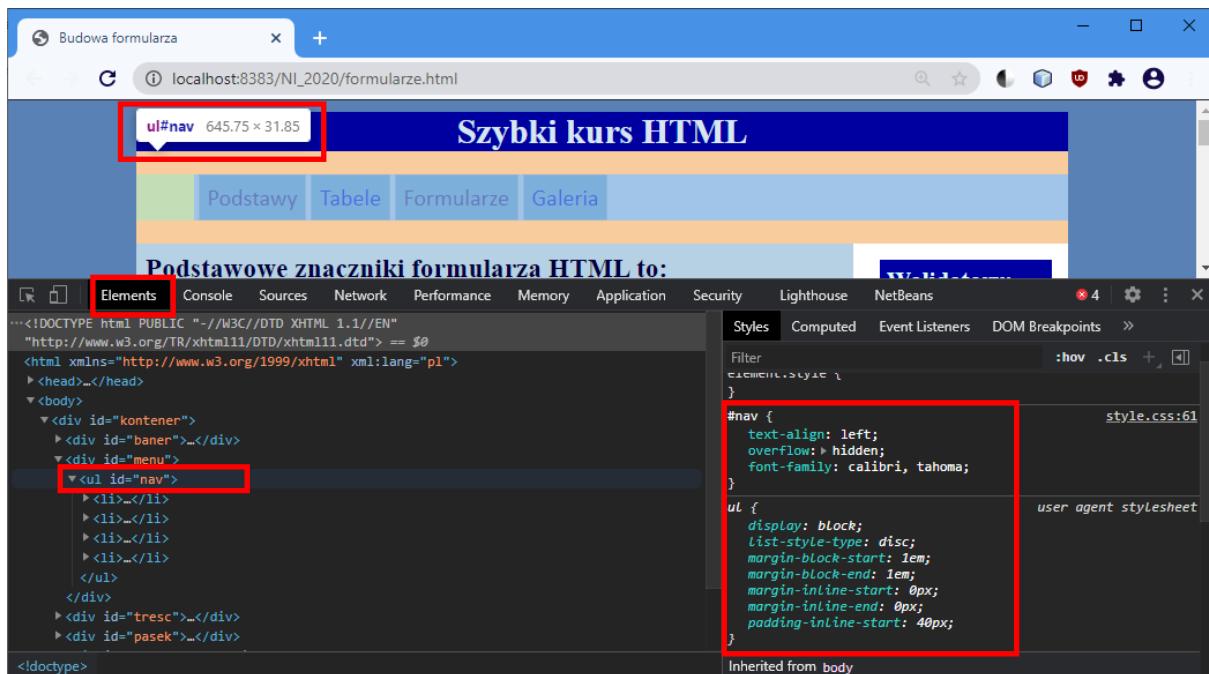
Korzystając z pliku *zdjecia.zip* (dostępne na platformie Moodle) utwórz stronę *galeria.html* (Rys. 3.6.). Każda miniaturka ma być linkiem do powiększonego zdjęcia, otwieranego (po kliknięciu na miniaturkę) w nowej karcie. Sprawdź wartości atrybutu *target* dla elementu *<a>*.

## Zintegrowany Program Rozwoju Politechniki Lubelskiej – część druga



Rys. 3.6. Przykładowa galeria zdjęć

Skorzystaj z narzędzi w przeglądarce w celu kontroli reguł CSS (Rys. 3.7).



Rys. 3.7. Widok strony i narzędzia deweloperskie w Chrome: zakładka **Elements** z możliwością podglądu struktury dokumentu i reguł CSS



Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska

Unia Europejska  
Europejski Fundusz Społeczny



**LABORATORIUM 4. POZNANIE ZAAWANSOWANYCH MOŻLIWOŚCI TWORZENIA FORMULARZY HTML5. REALIZACJA WALIDACJI DANYCH W POLACH FORMULARZY. METODY PRZESYŁANIA DANYCH Z FORMULARZA PO STRONIE KlientA NA SERWER WWW - WYKORZYSTANIE NARZĘDZI DOSTĘPNYCH W PRZEGŁĄDARCE. WSPARCIE PRZEGŁĄDAREK DLA PÓŁ FORMULARZY HTML5.**

**Cel laboratorium:**

Celem zajęć jest poznanie dodatkowych typów pól i ich atrybutów, które umożliwiają sprawdzenie poprawności danych po stronie przeglądarki, przed wysłaniem ich na serwer.

**Zakres tematyczny zajęć:**

Zbudowanie formularza z dodatkowymi typami pól i atrybutami do ich walidacji.  
Zastosowanie wyrażeń regularnych do kontroli poprawności danych.  
Przetestowanie wprowadzonych reguł walidacji - wykorzystanie wsparcia przeglądarek do sprawdzania poprawności danych.

**Pytania kontrolne:**

1. Podaj typy pól formularzy oraz ich atrybuty pomocne w procesie sprawdzania poprawności przesyłanych danych.
2. Co to jest wyrażenie regularne? Podaj przykłady.
3. Do czego służy atrybut *pattern* w polu *<input>*?

**Zadanie 4.1. Podstawowe pola i atrybuty walidacji formularzy HTML5**

- a) Utwórz dokument *walidacja.html*, który zawiera kod prostego formularza (Listing 4.1). Sprawdź poprawność tego kodu HTML korzystając z validatora W3C

**Listing 4.1.**

```
<!DOCTYPE html>
<html lang="pl">
    <head>
        <title> Formularz rejestracyjny</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <div>
            <form action="#" method="get">
                <p>
                    Imię*:<br />
                    <input type="text" name="imie" ><br />
                    Nazwisko*:<br />
                    <input type="text" name="nazwisko" ><br />
                </p>
            </form>
        </div>
    </body>
</html>
```



Fundusze Europejskie  
Wiedza Edukacja Rozwój



Rzeczypospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny

```
Email*:<br />
<input type="text" name="email"><br />
Data urodzenia:<br />
<input type="text" name="data"><br />
Hasło*:<br />
<input type="password" name="haslo"><br />
</p>
<p>
Uczeń: <input type="radio" name="kto" value="uczen">
Student: <input type="radio" name="kto" value="student">
Pracownik: <input type="radio" name="kto" value="pracownik"><br />
<br />
Zainteresowania:
<select name="zainteresowania" multiple size="3">
<option value="sport">Sport</option>
<option value="muzyka">Muzyka</option>
<option value="film">Film</option>
<option value="motoryzacja">Motoryzacja</option>
<option value="taniec">Taniec</option>
<option value="podróże">Podróże</option>
</select><br />
<input type="checkbox" name="akceptacja" >*Akceptuję regulamin <br />
<input type="submit" value="Rejestruj">
<input type="reset" value="Wyczyść">
</p>
</form>
</div>
</body>
</html>
```

- b) Otwórz formularz w przeglądarce (Rys. 4.1), wypełnij go i wyślij. Dane zebrane z pól formularza są przesyłane (**tylko** na potrzeby tego przykładu) za pomocą metody GET protokołu HTTP. W przypadku metody GET, parametry są dopisywane do adresu URL strony (sprawdź w przeglądarce, możesz też skorzystać z poznanych wcześniej narzędzi deweloperskich). Przeglądarka wyśle żądanie pod wskazany adres, nawet jeśli formularz nie zostanie wypełniony wcale lub dane nie będą wprowadzone we właściwym formacie.
- c) Aby, przynajmniej w pewnym stopniu, kontrolować poprawność wprowadzanych danych, uzupełnij pola formularza dodatkowymi atrybutami tak, aby uwzględnione były następujące wymagania (**testuj na bieżąco efekty wprowadzanych zmian** – Rys.4.2).
- wszystkie pola z symbolem \* powinny być oznaczone jako wymagane (wykorzystaj atrybut **required**),
  - użyj odpowiednich wartości atrybutu **type** (np. **email**, **date**, **tel** – tabela 4.1) dla wybranych pól formularza,
  - do pola z numerem telefonu oraz adresu e-mail dodaj podpowiedzi dotyczące formatu wprowadzanych danych za pomocą atrybutu **placeholder**,
  - dla pola imienia i nazwiska wykorzystaj atrybut **pattern**, tak aby w polu można było wprowadzić jedynie znaki liter. W imieniu uwzględnij możliwość wprowadzenia np. dwóch imion, a w nazwisku podania dwóch nazwisk z myślnikiem.



The screenshot shows a registration form titled "Formularz rejestracyjny". It contains fields for Name\*, Surname\*, Email\*, Date of Birth, and Password\*. Below these are radio buttons for "Uczeń", "Student", and "Pracownik". A dropdown menu for interests lists "Sport", "Muzyka", and "Film". There is also a checkbox for accepting the service terms and a note about required fields.

Imię\*:

Nazwisko\*:

Email\*:

Data urodzenia:

Hasło\*:

Uczeń:  Student:  Pracownik:

Zainteresowania:

\*Akceptuję regulamin serwisu

Rys. 4.1. Widok podstawowego formularza w Firefox

The screenshot shows the same registration form in two browsers: Chrome and Firefox. Both browsers highlight the "Telefon\*" and "Hasło\*" fields with red borders, indicating they are required. In Firefox, a tooltip "Proszę wprowadzić dane w żądanym formacie" appears over the "Hasło\*" field. The dropdown menu for interests is also highlighted with a red border.

Imię\*: Ania

Nazwisko\*: Anińska

Email\*: ania@gmail.com

Telefon\*: xyz

Podaj wartość w wymaganym formacie.

Uczeń:  Student:  Pracownik:

Zainteresowania:

\*Akceptuję regulamin serwisu

Rys. 4.2. Widok formularza w Chrome i Firefox z kontrolą poprawności danych



Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny

Podane w tabeli 4.1 znaczniki *input* można uzupełnić **dodatkowymi atrybutami** (w zależności od typu pola), np.:

- dla typu pola **number** jest możliwe określenie: najmniejszej wprowadzanej wartości – atrybut **min**, największej wprowadzanej wartości – atrybut **max**, skoku wartości za pomocą atrybutu **step**.
- dla typu pola **range** dostępne są atrybuty: **min**, domyślnie o wartości 0, **max**, domyślnie 100, **step**, domyślnie 1.

Tabela 4.1. Wybrane typy pól formularzy w HTML5

Typ	Opis
<input type="search">	jednowierszowe pole wyszukiwania
<input type="range">	suwak
<input type="number">	pole wartości liczbowej
<input type="color">	pole wyboru koloru
<input type="url">	pole adresu internetowego
<input type="email">	pole adresu e-mail
<input type="tel">	pole numeru telefonu
<input type="date">	pole wyboru daty
<input type="month">	pole miesiąca
<input type="time">	pole godziny
<input type="datetime">	pole wyboru daty i godziny

Więcej informacji na temat pól i atrybutów można znaleźć na stronach:

[https://www.w3schools.com/html/html\\_form\\_input\\_types.asp](https://www.w3schools.com/html/html_form_input_types.asp),

[https://www.w3schools.com/html/html\\_form\\_attributes.asp](https://www.w3schools.com/html/html_form_attributes.asp).

## Zadanie 4.2. Formularz dotacji

Stwórz kolejną stronę **dotacje.html** zawierającą formularz dotacji (Rys. 4.3) dla organizacji pozarządowej. Formularz powinien zawierać następujące pola wraz z wymaganiami:

- beneficjent, któremu przekazywana jest dotacja – pole tekstowe z **podpowiedziami** nazw organizacji (realizowane za pomocą **datalist**),
- wielkość dotacji – pole przyjmuje wartość liczbową nie mniejszą niż 20, ustawiony krok zmiany wielkości dotacji z wykorzystaniem atrybutu **step**,
- rodzaj karty płatniczej – pole typu **radio**,
- numer karty – pole tekstowe z walidacją poprawności numeru karty za pomocą atrybutu **pattern**. Walidacja ma obejmować przynajmniej weryfikację, czy numer karty złożony jest z 13 lub 16 cyfr.
- data ważności karty – pole uwzględniające miesiąc i rok ważności karty,
- kod CVV – pole złożone z 3 lub 4 cyfr z walidacją poprawności za pomocą atrybutu **pattern**,
- darczyńca – pole tekstowe o długości co najmniej 3 znaków weryfikowanych za pomocą atrybutu **pattern**.

Wszystkie pola formularza są **wymagane** i **dodatkowo** powinny posiadać atrybut **title** ze wskazówkami uzupełnienia pola (rys. 4.3).



Fundusze Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny

Na koniec pogrupuj pola formularza za pomocą znacznika **<fieldset>** i opisz odpowiednią grupę z zastosowaniem znacznika **<legend>** (Rys. 4.3).

The screenshot shows a web browser window titled "Dotacja" with the URL "localhost:8383/War...". The main content is a "Formularz dotacji" (Donation Form). It contains the following fields:

- Beneficjent: Wioski Dziecięce SOS
- Wartość dotacji: 25
- Dane karty płatniczej:
  - Karta:  Visa  Master Card
  - Numer: 1234567890123223
  - Data ważności: grudzień 2021
  - Kod CVV: 12
- Darczyń
  - Darek**
  - Rejestru**

A tooltip is displayed over the "Kod CVV" field, containing the following text:

! Podaj wartość w wymaganym formacie.  
Podaj kod CVV złożony z 3 lub 4 cyfr podanych na odwrocie karty

Rys. 4.3. Przykładowa realizacja formularza dotacji – widok w Chrome



## LABORATORIUM 5. REALIZACJA RESPONSYWNEJ STRONY INTERNETOWEJ Z WYKORZYSTANIEM ZAPYTANIA O MEDIA

### Cel laboratorium:

Celem zajęć jest wykorzystanie zapytań o media do stworzenia responsywnej strony internetowej.

### Zakres tematyczny zajęć:

Zbudowanie testowej strony internetowej w oparciu o dodatkowe elementy struktury.

Zdefiniowanie arkusza reguł CSS z zastosowaniem zapytań o media.

Przetestowanie responsywności strony.

### Pytania kontrolne:

1. Jakie znasz elementy struktury strony HTML5?
2. Co to jest zapytanie o media i w jaki sposób się je definiuje?
3. Rozwiń skrót RWD i wyjaśnij na czym polega responsywność stron internetowych.

### Zadanie 5.1. Dokument HTML5 z elementami struktury strony

Utwórz dokument *rwd.html* z zastosowaniem dodatkowych elementów struktury strony dodanych w standardzie HTML5 (*header*, *footer*, *section*, *nav* itp.) postaci jak na listingu 5.1.

#### Listing 5.1. Przykładowa strona z wykorzystaniem elementów struktury HTML5

```
<!DOCTYPE html>
<html>
  <head>
    <title>Przykład RWD</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <link rel="stylesheet" href="CSS/rwd.css" />
  </head>
  <body>
    <header>
      <h1>Przykład strony responsywnej</h1>
      <nav>
        <a href="index.html">Link 1</a>
        <a href="strona2.html">Link 2</a>
        <a href="strona3.html">Link 3</a>
        <a href="strona4.html">Link 4</a>
      </nav>
    </header>

    <section>
      <h2>O RWD</h2>
      <p>
        Projektowanie responsywne (RWD)
        ...
      </p>
    </section>
  </body>
</html>
```



Fundusze Europejskie  
Wiedza Edukacja Rozwój



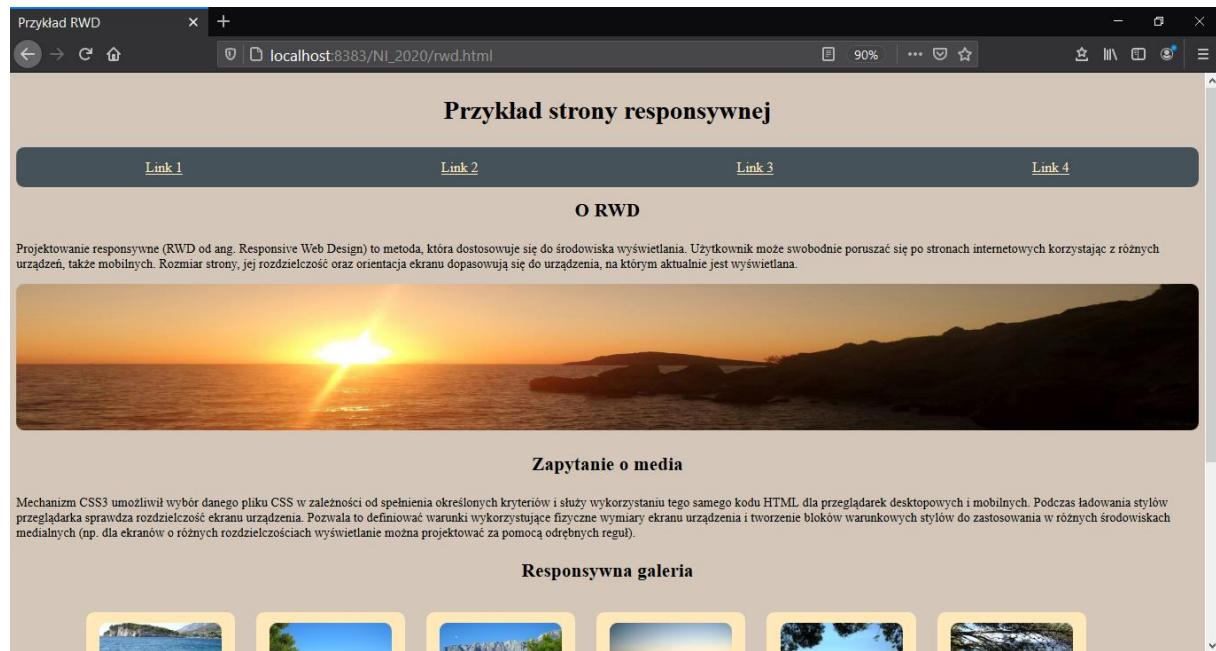
Rzeczypospolita  
Polska



```
</p>

<p class="srodek">
    
</p>
</section>
<section>
    <h2>Zapytanie o media</h2>
    <p>
        Mechanizm CSS3 umożliwił wybór danego pliku CSS ...
    </p>
</section>
<section>
    <div class="galeria">
        <h2>Responsywna galeria</h2>
        <div class="slajd">
            
        </div>
        <div class="slajd">
            
        </div>
    </div>
</section>
<footer>&copy;BP</footer>
</body>
</html>
```

Naszym zadaniem będzie dostosowanie strony dla urządzeń o różnych szerokościach wyświetlaczów. Pierwszy widok, do którego dążyliśmy, przedstawiony został na rys. 5.1a i 5.1b.



Rys. 5.1a. Widok strony bez zapytania o media na dużym ekranie



Rys. 5.1b. Widok strony bez zapytania o media przy mniejszej szerokości okna przeglądarki

W widoku mobilnym bardzo istotna jest rola znacznika:

<meta name="viewport"

content="width=device-width, initial-scale=1.0, user-scalable=true">

który określa, że dany dokument został zoptymalizowany z myślą o urządzeniach mobilnych.  
W atrybucie **content** ustawić dyrektywy:

- **width=device-width** - określa, że szerokość obszaru wyświetlania powinna odpowiadać szerokości ekranu danego urządzenia;
- **initial-scale** - ustawia początkową wartość skalowania (przybliżenia) dla danej strony (domyślana wartość początkowa zależy od przeglądarki stosowanej na urządzeniu; wartość 1.0 powoduje wyświetlenie dokumentu bez skalowania);
- **user-scalable** - ustala, czy użytkownik może przybliżać i oddalać obszar wyświetlania.

Kolejny etap to przygotowanie arkusza stylów do odpowiedniego wyświetlania elementów strony. Pierwszą rzeczą, którą możemy zrobić, jest sformatowanie obrazka jako **fluid image**, czyli nakazanie przeglądarce płynnego dostosowywania rozmiaru obrazka do podanej szerokości, co realizuje prosta reguła CSS:

img { max-width: 100%; height: auto; }

Przykładowy arkusz CSS może być postaci jak na listingu 5.2.

#### Listing 5.2. Pierwszy arkusz CSS

```
body { background:#d4c7b9; }
h1,h2,footer { padding:0.2em; text-align:center; }
```



Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



Rzeczypospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny

```
.srodek { text-align:center; }
nav{ float:left; width:100%; }
nav, footer {background: #45525a; color:#ffe9ba; border-radius:10px;}
nav a {
    float:left;
    display:block;
    width:23%;
    padding:1%;
    font-size:110%;
    color:#ffe9ba;
    text-align:center;
}
section { clear:both;}
img { max-width: 100%; height: auto; border-radius:10px; }
.slajd {
    float:left;
    display:block;
    width:13%;
    text-align:center;
    border-radius:10px;
    background:#ffe9ba;
    padding:1% 0.5%;
    margin:1%;
}
.slajd:hover { background:#45525a; }
footer { clear:both; }
.galeria{ margin:auto; width:90%;}
```

### Zadanie 5.2. Zapytania o media

Kolejnym zadaniem jest zadeklarowanie innych reguł formatowania dla poszczególnych rozmiarów okien za pomocą tzw. **zapytania mediów** (ang. media queries).

Zapytanie o media może mieć postać jak na listingu 5.3.

### Listing 5.3. Zapytanie o media

```
@media screen and (max-width: 720px) {
    nav a { width: 48%;
            padding: 10px 0;
            font-size:140%;}
    .slajd {width: 45%; }
}
@media screen and (min-width: 481px) and (max-width: 720px) {
    body { font-size: 13px; }
    h1 { font-size: 23px; }
    h2 { font-size: 18px; }
}
```

Po dodaniu powyższych reguł na koniec do arkusza *style.css*, przy odpowiednio małej szerokości okna przeglądarki – wynik będzie jak na rysunku 5.2.

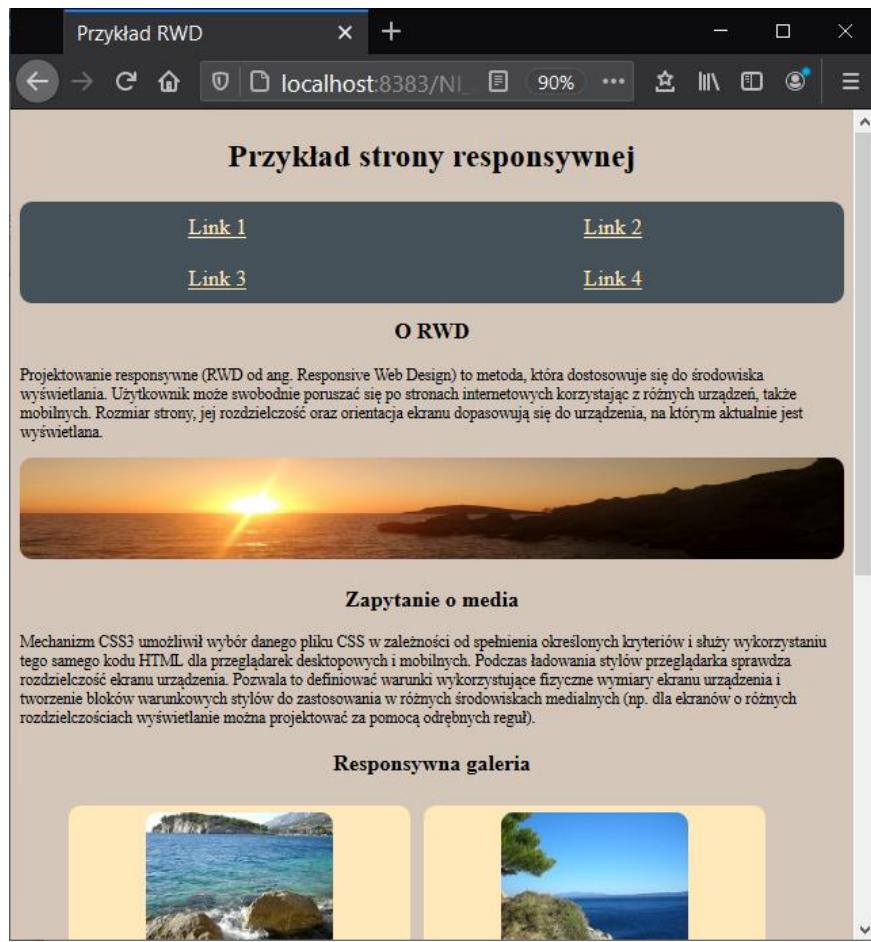


Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska





Rys. 5.2. Widok strony po dodaniu zapytania o media

### UWAGA

Na listingu 5.3 typem medium w obu przypadkach jest **screen**, czyli ekran komputera. Istnieją również takie typy medium jak syntezator mowy, materiały drukowane nieprzezroczyste, odbiorniki TV, czy też drukarki Braille. Więcej informacji na temat mediów można znaleźć na stronie: [http://www.w3schools.com/css/css\\_mediatypes.asp](http://www.w3schools.com/css/css_mediatypes.asp)

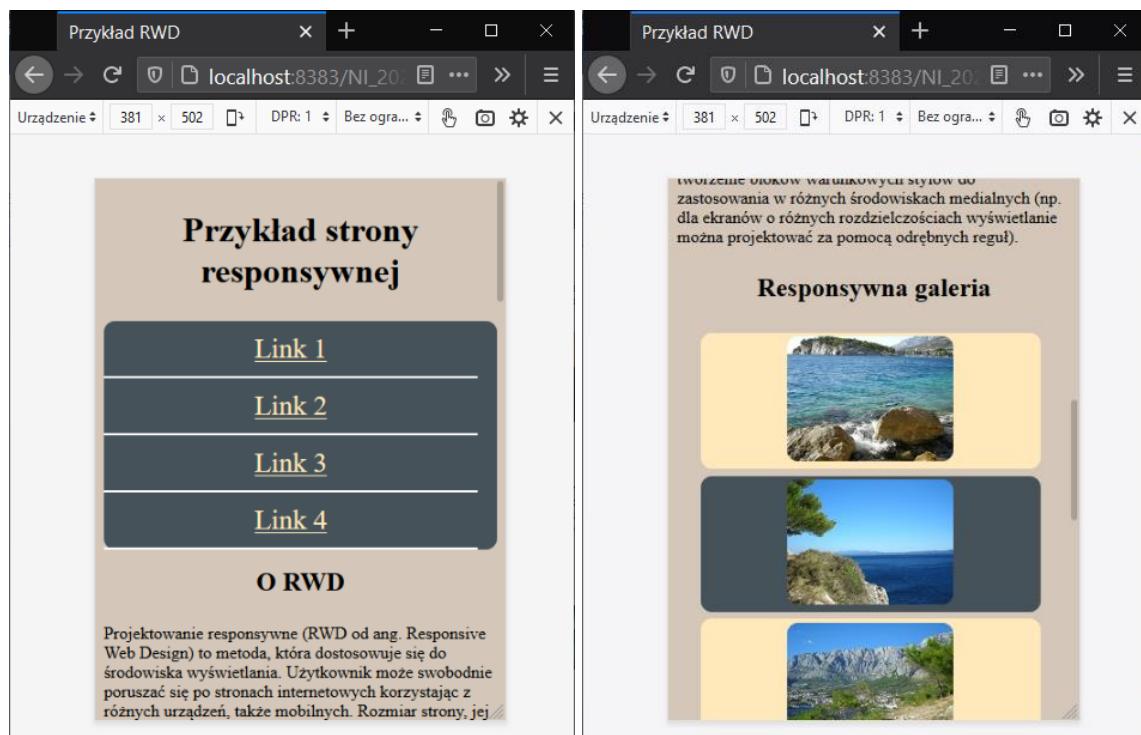
### Zadanie 5.3. Dodatkowe zapytania o media

Do reguł CSS dodaj kolejne zapytanie o media, tak aby przy szerokościach mniejszych niż **481 px** strona renderowała się podobnie jak na rysunku 5.3a i 5.3b. Testowanie widoku responsywnego oferują narzędzia dostępne w przeglądarkach, np. Chrome (ikonka widoku mobilnego na pasku narzędzi deweloperskich), czy Firefox (*Narzędzia -> Dla twórców witryn -> Tryb responsywny*).

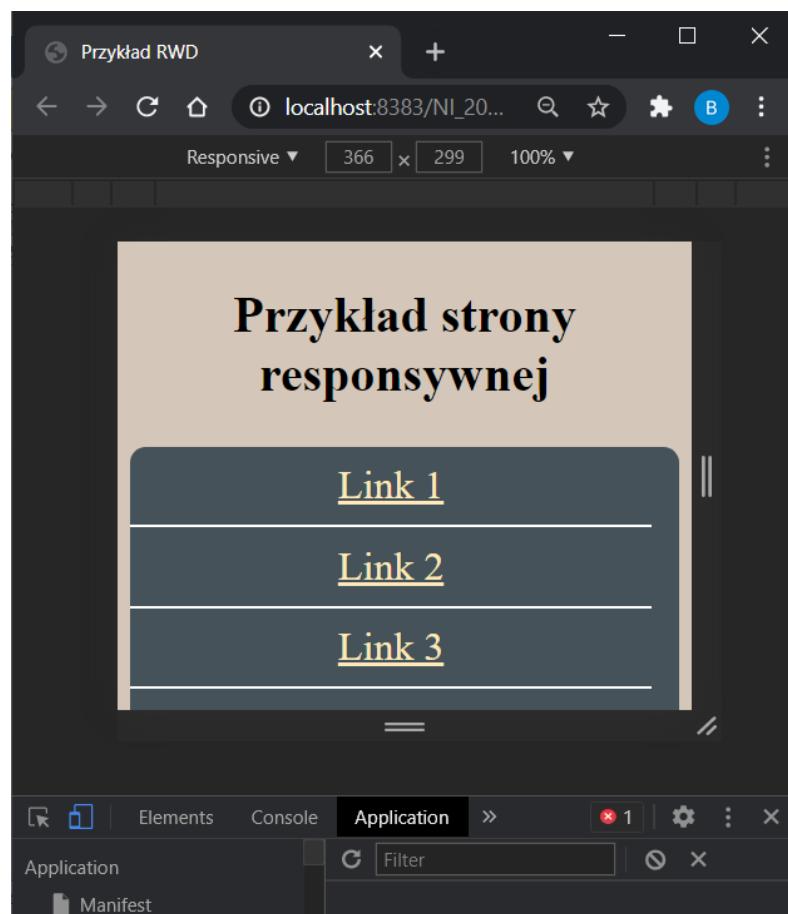
### UWAGA

W celu symulacji renderowania strony na urządzeniach o różnych szerokościach można również skorzystać ze strony: <http://testrwd.pl/>

Przykładowe widoki symulowane na powyższej stronie przedstawia rysunek 5.4.



Rys. 5.3a. Widok strony po dodaniu zapytania o media <481px w widoku mobilnym w Firefox



Rys. 5.3b. Widok strony po dodaniu zapytania o media <481px w widoku mobilnym w Chrome



Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska

Unia Europejska  
Europejski Fundusz Społeczny





Rys. 5.4. Widok strony na symulatorze testrwd

Korzystając z zapytania o media można również ustalić oddzielne reguły dla urządzeń trzymanych w pozycji wertykalnej (**portrait**) i horyzontalnej (**landscape**). Przykład wykorzystujący orientację został pokazany na listingu 5.4.



Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska

Unia Europejska  
Europejski Fundusz Społeczny



**Listing 5.4. Przykład reguł z uwzględnieniem orientacji urządzenia**

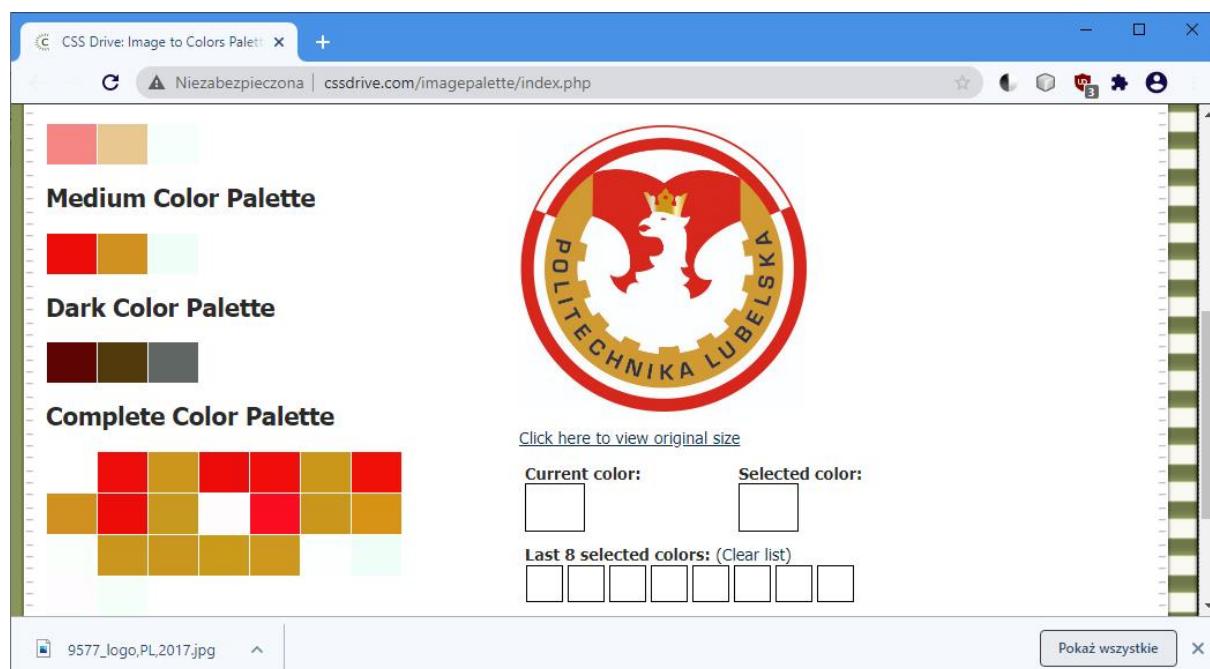
```
@media only screen and (min-width : 768px)
    and (max-width : 1024px)
    and (orientation : landscape)
{
    /* Reguły CSS poszczególnych elementów strony */
}
```

**Zadanie 5.4. Orientacja pionowa i pozioma**

Do reguł CSS dodaj kolejne zapytanie o media, tak aby zróżnicować wygląd strony w zależności od orientacji urządzenia.

**Ciekawe strony pomocne do realizacji projektów:**

- Do wyboru palety kolorów (dopasowanych np. do kolorystyczki zdjęcia, loga) można skorzystać z narzędzia dostępnego na stronach np.:
  - <http://www.cssdrive.com/imagepalette/> (Rys. 5.5),
  - <http://www.colorhunter.com/> (Rys. 5.6).
- W celu wstawienia „czcionki” jako logo stron serwisów społecznościowych można skorzystać ze strony:
  - <http://www.fontello.com/> (Rys. 5.7).



Rys. 5.5. Dobór palety kolorów na podstawie logo PL



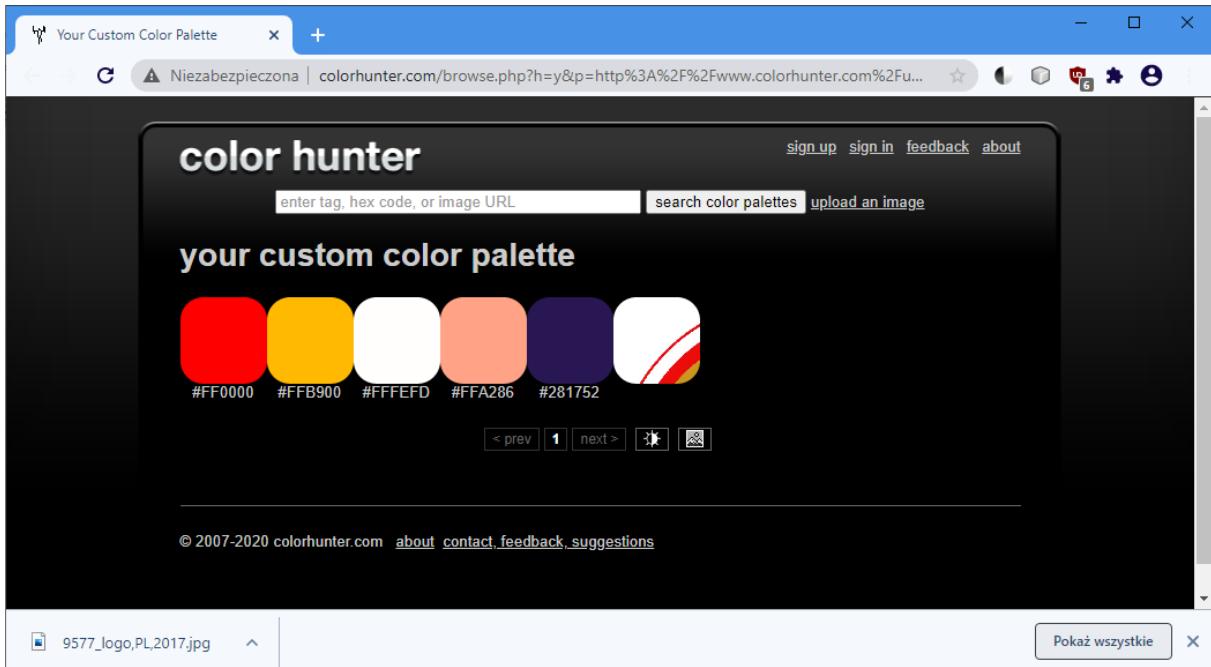
Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



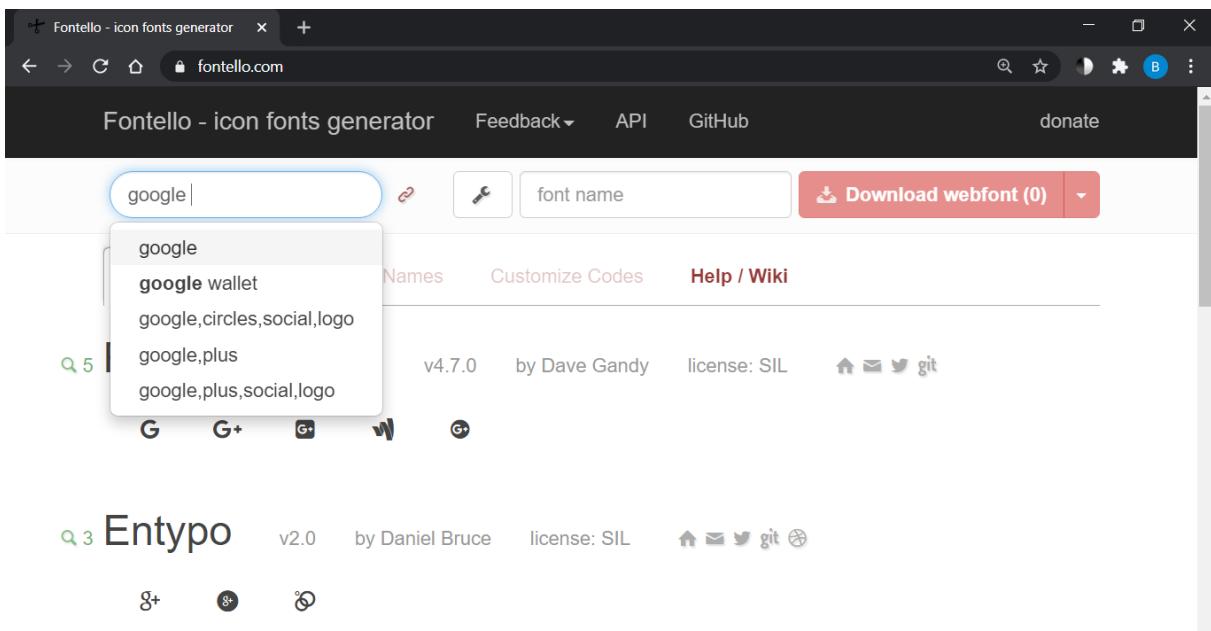
Rzeczpospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny



Rys. 5.6. Dobór palety kolorów na podstawie loga PL



Rys. 5.7. Strona Fontello z możliwością pobrania definicji fontów

#### Zadanie 5.5. Kontrola reguły CSS za pomocą przeglądarki

Z pomocą narzędzia wbudowanego w przeglądarkę sprawdź reguły CSS ustalone dla elementów strony *rwd.html*. W tym celu w przeglądarce Chrome wystarczy, po wskazaniu myszą dowolnego elementu na stronie (np. nawigacji), skorzystać z menu kontekstowego i wybrać opcję **Zbadaj** lub w opcjach narzędzi developerских wskazać zakładkę **Elements**. Dla wskazanego elementu zostanie wyświetlony *Box Model*, który pozwoli sprawdzić jakie reguły zostały mu przypisane (również te domyślne) (Rys. 5.8). Wartości deskryptorów CSS



Fundusze  
Europejskie  
Wiedza Edukacja Rozwój

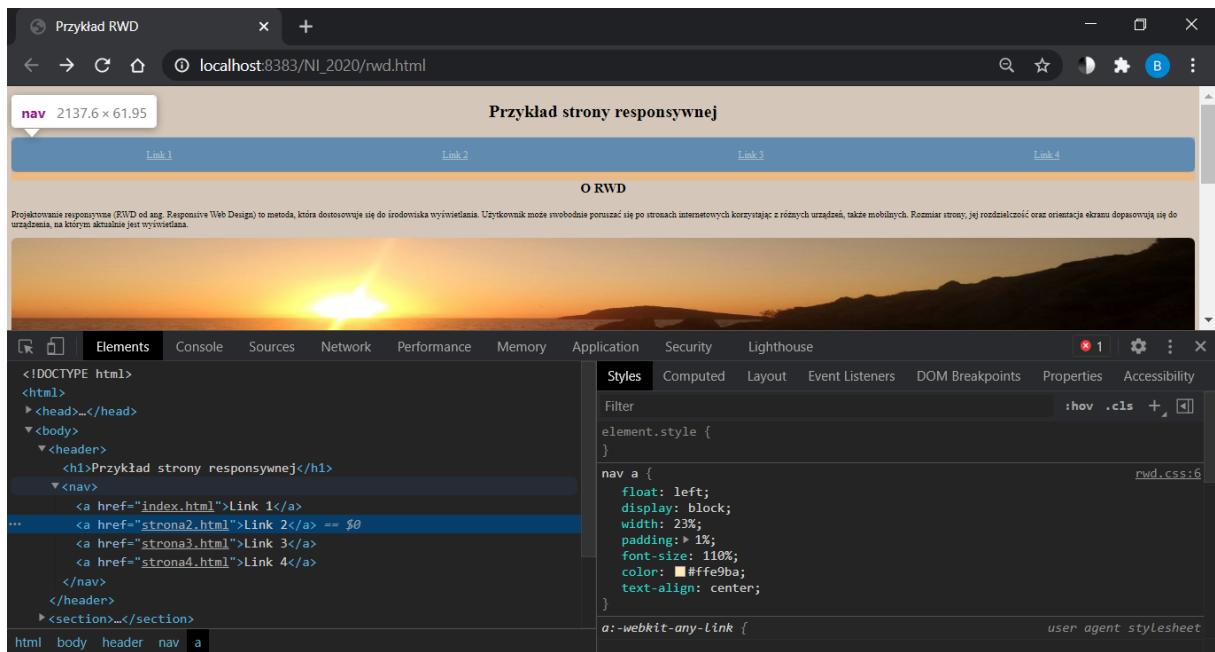


Rzeczpospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny

można dowolnie modyfikować i obserwować efekt bezpośrednio na stronie w przeglądarce. Jest to bardzo przydatne w fazie projektowania strony wizualnej serwisu.



Rys. 5.8. Kontrola reguł CSS w przeglądarce Chrome

### Zadanie 5.6. Modyfikacje w projekcie

Utwórz kopię projektu zrealizowanego na laboratoriach 1-3, a następnie zmodyfikuj dokumenty HTML i CSS, tak, aby zastosować dodatkowe elementy struktury strony HTML, pola i atrybuty dla formularzy, dopasować odpowiednio reguły CSS.

W tym celu:

- zamiast bloków `<div id="...">` zastosuj odpowiednie elementy struktury dokumentu HTML5 (np. `<header>`, `<nav>`, `<aside>`, `<footer>` itp.);
- dopasuj istniejące już reguły CSS do nowych elementów (np. selektor `#baner` zastąp selektorem `header`, selektor `#stopka` zastąp selektorem `footer`, itp.). Dopasuj kolorystykę strony do własnych preferencji;
- uzupełnij reguły CSS o dodatkowe możliwości według uznania (np. zaokrąglone narożniki - **`border-radius`**, przezroczystość elementów - **`box-shadow`**), dla pliku **`tabele.html`** – uprość reguły formatowania parzystych i nieparzystych wierszy tabeli – zastosuj **`td:nth-child`**, itp.;
- wykorzystaj właściwość **`flexbox`** zamiast **`float`** do pozycjonowania elementów w bloku nawigacji oraz do utworzenia układu kolumnowego. Zastosuj też **`box-sizing`**;
- dodaj przynajmniej jedno zapytanie o media **`@media`**, tak aby układ dwukolumnowy przechodził w jedną kolumnę dla wskazanej szerokości (np. `<720px`);
- na stronie **`formularze.html`** zmodyfikuj już istniejące pola `<input>` dodatkowymi wartościami dla atrybutu `type` (np. `email`, `number`) i dodatkowymi atrybutami do validacji (np. `min`, `max`, `required`, `placeholder`, `title`, `pattern`). Sprawdź efekt działania w przeglądarce po wpisaniu błędnych danych;
- przeprowadź validację kodu zmodyfikowanych plików `.html` i arkuszy CSS. Skorzystaj z validatorów dla HTML i CSS.



## LABORATORIUM 6. WPROWADZENIE ELEMENTÓW INTERAKCJI

**Z UŻYTKOWNIKIEM W APLIKACJI INTERNETOWEJ ZA  
POMOCĄ JĘZYKA SKRYPTOWEGO JAVASCRIPT.  
REALIZACJA OBSŁUGI ZDARZEŃ NA STRONIE WWW  
Z ZASTOSOWANIEM MODELU DOM. KORZYSTANIE  
Z KONSOLI JAVASCRIPT DOSTĘPNEJ W PRZEGŁĄDARCE  
INTERNETOWEJ.**

### Cel laboratorium:

Celem zajęć jest wykorzystanie JavaScript do obsługi podstawowych zdarzeń zachodzących na graficznym interfejsie użytkownika strony internetowej.

### Zakres tematyczny zajęć:

Dodanie do projektu z poprzednich laboratoriów kolejnych stron wykorzystujących JavaScript.

Realizacja obsługi zdarzeń na stronie z zastosowaniem modelu DOM.

Zastosowanie konsoli JavaScript do testowania działania skryptów.

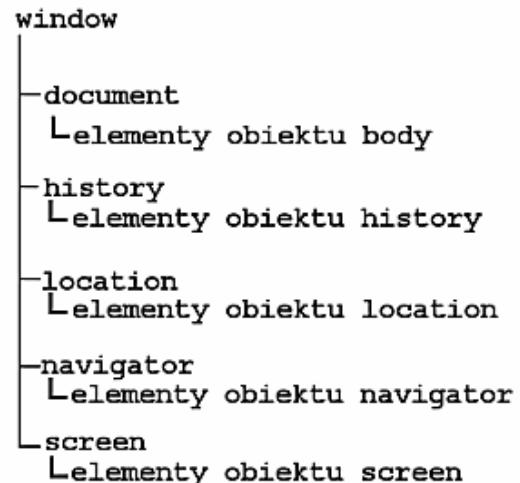
### Pytania kontrolne:

1. Jakie znasz sposoby osadzania skryptów JS w dokumencie HTML?
2. Jakie atrybuty elementów dokumentu HTML można zastosować do obsługi zdarzeń?
3. Co to jest model DOM?
4. Podaj podstawowe metody JavaScript, które umożliwiają pracę z DOM.

### Wprowadzenie do JavaScript

CSS i JavaScript używają modelu DOM do operowania na elementach dokumentu HTML/XML. Według W3C, **model DOM** to drzewiasty zbiór węzłów. Węzeł może być **elementem, ciągiem tekstu, komentarzem**, itp. Każdy węzeł może mieć jednego rodzica, dowolną liczbę braci (lub sąsiadów) i, jeśli jest **elementem**, dowolną liczbę dzieci.

JavaScript jest językiem obiektowym – prawie wszystkie funkcje, z których korzystamy, są wywoływanie z jakiegoś obiektu. Przeglądarka udostępnia nam np. obiekt **window**, który reprezentuje okno zawierające dokument (obiekt **document**), który jest załadowany w tym oknie (Rys. 6.1). Z obiektu **document** możemy wywoływać różne przydatne funkcje (metody), umożliwiające dostęp do dowolnego elementu na stronie, a także modyfikacje jego zawartości czy atrybutów.



Rys.6.1. Niektóre obiekty dostępne z poziomu okna window w przeglądarce

W JS odwołanie się do danego elementu, który posiada unikatowy atrybut **id** odbywa się za pomocą metody:

```
element = document.getElementById('id');
```

Metoda ta zwraca **element** HTML np. *body*, *div*, *p*, *td*, *input*). Do pobrania lub ustalenia zawartości elementu (jeśli posiada on zawartość), służy właściwość:

```
element.innerHTML = zmienienna_typu_string;
```

Aby pobrać/ustalić wartość pola formularza, korzystamy z właściwości:

```
tekst = element.value; //pobranie tekstu z pola formularza
```

```
element.value = zmienienna; //ustawienie wartości zmiennej w polu
```

Interakcja z użytkownikiem jest realizowana za pomocą obsługi zdarzeń, które mogą zachodzić na elementach dokumentu (np. zdarzenia związane z kliknięciem przycisku, przemieszczaniem się myszy nad elementem, wysłanie formularza, aktywacja/dezaktywacja elementu itp.). Najprościej zdarzenia takie są obsługiwane za pomocą wywołania odpowiedniej funkcji JS, której nazwa jest podana jako wartość atrybutu HTML do obsługi odpowiedniego typu zdarzenia. Atrybuty obsługi zdarzeń są zawsze poprzedzone prefiksem **on** z nazwą zdarzenia, np.:

```
<p onclick="funkcja_js()".>
```

### UWAGA

JavaScript rozróżnia duże i małe litery, obowiązuje następująca konwencja nazw: wszystkie metody zaczynają się małą literą, ale poszczególne wyrazy w nazwie metody pisane są już z wielkiej litery, np. **getElementById**.

### Zadanie 6.1. Prosty formularz obliczeniowy

Przykładowa strona ze skryptem JS generuje obraz przedstawiony na rys. 6.2 – funkcja *oblicz()* wykorzystuje dostęp do elementów HTML za pomocą metody **getElementById**. Metoda ta pozwala pobrać **elementy** dokumentu HTML o wskazanym atrybutem **id**.

Należy:

- a) uruchomić i przeanalizować działanie skryptu osadzonego wewnętrz dokumentu *js1.html* (w części nagłówkowej **<head>**):

```
<!DOCTYPE ...>
<head>
    <meta ... />
    <title> Dostęp do pól formularza</title>
    <script type="text/javascript">
        function oblicz()
        { //Słówko var - oznacza zmieniąną; typ tej zmiennej
            //będzie zależał od jej wartości;
            //Pobierz element o id=l1 do zmiennej l1:
            var l1=document.getElementById('l1');
            //pobierz wartość (łańcuch tekstowy) wpisaną w polu
            //formularza o id='l1':
            l1=l1.value;
            //Przekonwertuj (jeśli się uda) l1 do wartości typu int:
            l1=parseInt(l1);
            //Jeśli udała się próba konwersji to l1 zawiera wartość całkowitą
            //Analogicznie realizujemy pobranie wartości z drugiego pola
            // tekstopowego, ale tym razem wszystko w jednej instrukcji:
            var l2=parseInt(document.getElementById('l2').value);
```



Fundusze Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska

Unia Europejska  
Europejski Fundusz Społeczny



```
//Teraz już możemy obliczyć sumę i ustawić wartość pola tekstowego
//o id='suma':
var s=document.getElementById('suma');
s.value=l1+l2;
}
</script>
</head>
<body> <div><h3>Dodawanie</h3>
<!-- Elementy formularza (pole tekstowe i przycisk) zostały umieszczone
w polach tabeli; każde pole formularza posiada unikatowe id, dzięki
któremu możemy w JavaScript zastosować metodę getElementById -->
<table><tbody>
<tr><td>Liczba 1:</td><td><input id="l1" /></td> </tr>
<tr><td>Liczba 2:</td><td><input id="l2" /></td> </tr>
<tr><td><button onclick="oblicz()">Oblicz sumę : </button></td>
<!-- Pole wynikowe suma posiada dodatkowy atrybut disabled, dzięki
czemu tylko metoda obliczeniowa w JavaScript ma możliwość ustawiania
wyniku - dla użytkownika strony pole jest niedostępne -->
<td><input id="suma" disabled value="" /></td></tr>
</tbody></table></div>/
</body>
</html>
```

Zauważ, że w HTML pola formularza nie są teraz ujęte w blok `<form>` - są one wykorzystywane tylko do pobrania i ustawienia na nich wartości po stronie klienta (przeglądarki), nie są wysyłane na serwer, wobec czego nie ma potrzeby stosowania znacznika `<form>`.

- b) zmodyfikować stronę tak, aby korzystała ze skryptu umieszczonego w zewnętrznym pliku - w tym celu należy cały kod funkcji:

```
function oblicz() { ... }
```

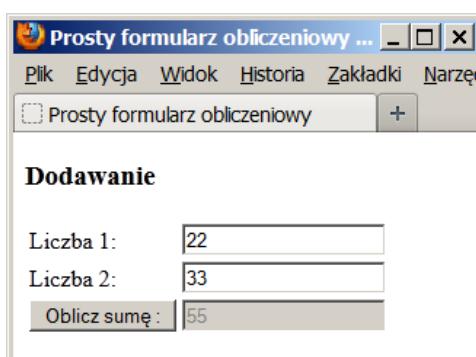
zapisać w pliku, np. `funkcje.js`, a następnie w dokumencie HTML wymienić fragment:

```
<script type="text/javascript">
    function oblicz(){ ... }
</script>
```

na (atribut `type` można pominąć):

```
<script src="funkcje.js"></script>
```

- c) sprawdzić czy wszystko działa poprawnie.



Rys.6.2. Prosty formularz obliczeniowy



Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska

Unia Europejska  
Europejski Fundusz Społeczny



Naszym zadaniem będzie teraz dodanie do projektu z poprzednich laboratoriów skryptów JavaScript.

Dla przypomnienia, struktura projektu z laboratoriów 1-3, uzupełniona o folder na pliki JS powinna wyglądać następująco:

- WWW - *index.html*
- *tabele.html*
- *formularze.html*
- CSS - *style.css*
  - *tabele.css*
  - *formularze.css*
- JS - *funkcje.js*

### Zadanie 6.2. Formularz obliczeniowy w projekcie

Na podstawie zadania 6.2 przygotuj skrypt JS do obliczenia raty przy spłacie pożyczki w równych ratach miesięcznych (rys. 6.3), korzystając z wzoru:  $rata = \frac{K * pr\_mc}{1 - \frac{1}{(1 + pr\_mc)^n}}$ ,

gdzie:  $K$  - kwota pożyczki,

$pr$  - oprocentowanie roczne,

$n$  - liczba rat,

$pr\_mc=pr/12$  – oprocentowanie w skali miesiąca.

Na początek przygotuj i przetestuj działanie samego formularza do obliczeń. Docelowo gotowy formularz umieść na stronie *obliczenia.html*, której struktura powinna bazować na szablonie z pliku *index.html*. Zwróć uwagę na rozwijaną opcję menu *Obliczenia w formularzu* z linkiem do nowej strony *obliczenia.html*, który pojawił się w menu (Rys.6.3). Pamiętaj o dodaniu w części nagłówkowej dokumentu znacznika:

```
<script src="JS/funkcje.js"></script>
```

W przypadku wprowadzenia błędnych danych - powinien być wyświetlony odpowiedni komunikat (nie powinna się wyświetlić wartość specjalna *Infinity* lub *NaN*). Do sprawdzenia wartości specjalnych wykorzystaj funkcje: *isNaN()*, *isFinite()*.

### Zadanie 6.3. Operacje arytmetyczne

Utwórz formularz HTML postaci jak na rysunku 6.4 a następnie zaimplementuj funkcję JS do wykonania wskazanego przez przycisk radio działania. Pamiętaj, że wszystkie przyciski **radio w grupie** muszą posiadać tę samą wartość dla atrybutu *name* i jednocześnie różne wartości dla atrybutu *value* oraz, że metoda *document.getElementsByName()* zwraca **tablicę** elementów dokumentu HTML o wskazanej wartości atrybutu *name*. W przypadku wprowadzenia błędnych danych lub próby dzielenia przez 0 - powinien być wyświetlony odpowiedni komunikat (nie powinna się wyświetlić wartość specjalna *Infinity* lub *NaN*).

Fragment funkcji *oblicz()* może wyglądać jak na listingu 6.1.

#### Listing 6.1. Schemat funkcji kalkulatorka

```
function oblicz(){  
    var tab = document.getElementsByTagName("operator");  
    var op; //operacja arytmetyczna do wykonania
```



Fundusze Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska

Unia Europejska  
Europejski Fundusz Społeczny



```
for(let i=0;i<tab.length;i++){
    if(tab[i].checked) op = tab[i].value;
}
//pobierz x
//pobierz y
//oblicz i pokaż wynik w zależności od operatora (zastosuj switch)
}
```

Po przetestowaniu działania skryptu - dodaj stronkę z kalkulatorkiem do menu rozwijanego z opcji Formularze w projekcie.

The screenshot shows a web browser window titled "Formularz obliczeniowy". The address bar indicates the URL is "localhost:8383/NI\_2020/obliczenia.html". The main content area has a blue header "Szybki kurs HTML" and a navigation bar with tabs: "Podstawy", "Tabele", "Formularze" (which is highlighted), and "Galeria". A dropdown menu is open over the "Formularze" tab, with "Obliczenia w formularzu" selected. Below the dropdown, there is a form for calculating a loan ("Pożyczka") with fields for Kwota (1000.0), Ile rat (10), Procent (10.0), Rata miesięczna (104.64), and Kwota z odsetkami (1046.40). A button labeled "Oblicz ratę stałą" is present. To the right, there are two boxes: "Validatory" listing "HTML" and "CSS", and "Kursy" listing "HTML", "CSS", "JS", and "PHP". The footer contains the text "©BP".

Rys.6.3. Dodatkowy formularz do obliczenia raty w projekcie

The screenshot shows a web browser window titled "Kalkulator". The address bar indicates the URL is "localhost:8383/NI\_2020/obliczenia.html". The main content area has a title "Kalkulator". It contains two sections: "Wpisz liczby:" and "Wskaż działanie:". In "Wpisz liczby:", there are input fields for "X" (containing "100.5") and "Y" (containing "0.5"). In "Wskaż działanie:", there are radio buttons for "+", "-", "\*", and "/". Below these, a "Oblicz" button is visible, and next to it is a "Wynik:" field containing "201".

Rys. 6.4. Formularz kalkulatora



Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny

## LABORATORIUM 7. BUDOWA PROSTEJ APLIKACJI TYPU SPA Z WYKORZYSTANIEM FUNKCJI JAVASCRIPT.

### Cel laboratorium:

Celem zajęć jest wykorzystanie już nabytych umiejętności tworzenia stron responsywnych do budowy prostej aplikacji typu SPA z wykorzystaniem funkcji JavaScript.

### Zakres tematyczny zajęć:

Wykorzystanie elementów projektu responsywnego stworzonego na laboratorium 5 do budowy aplikacji jednostronicowej.

Obsługa nawigacji na stronie za pomocą funkcji JavaScript.

Generowanie treści strony za pomocą funkcji JavaScript.

Zastosowanie JavaScript do validacji danych z formularza.

### Pytania kontrolne:

1. Wyjaśnij skrót SPA.
2. Jak można wykorzystać właściwość **innerHTML** elementu dokumentu HTML?
3. Jak można wykorzystać właściwość **value** elementu pola formularza?
4. W jaki sposób można, za pomocą funkcji JS, pobrać dane z tekstuowego pola formularza? Podaj przykład.

### Zadanie 7.1. Prosta aplikacja typu SPA

Utwórz dokument HTML5 (Listing 7.1), podobny jak na laboratorium 5 tyle, że tym razem zamiast nawigacji opartej na hiperlinkach, nawigacja będzie realizowana za pomocą przycisków (element *button*), które będą obsługiwane przez odpowiednie funkcje JavaScript. Zadaniem funkcji JS będzie modyfikacja zawartości elementu o **id="blok"**, w zależności od tego, który przycisk wybierze użytkownika. Będziemy teraz pracować z jednym dokumentem HTML (zamiast tworzyć 4 różne dokumenty, jak było to realizowane poprzednio).

Zwróć uwagę na znacznik:

```
<script src="js/funkcje.js"></script>
```

który dołącza skrypt JS z definicjami funkcji, wykorzystywanych do obsługi przycisków (plik **funkcje.js** znajduje się w folderze **js**).

Listing 7.2 przedstawia kod pliku **style.css**.

#### Listing 7.1. Plik index.html z przyciskami akcji w elemencie nav

```
<!DOCTYPE html>
<html>
  <head>
    <title>Przykłady Java Script</title>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" href="css/style.css" />
    <script src="js/funkcje.js"></script>

  </head>
  <body>
    <header>
```



Fundusze Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny

```
<h1>Aplikacja SPA</h1>
<nav>
    <button onclick="pokaz(1)">Pierwsze kroki </button>
    <button onclick="pokaz(2)">Galeria</button>
    <button onclick="pokaz(3)">Dodaj post</button>
    <button onclick="pokaz(4)">Kontakt</button>
</nav>
</header>
<section id="blok">
    <h2><br />Pierwsze kroki</h2>
    <p>
        W aplikacjach typu SPA (ang. Single Page Application) ...
    </p>
    <p class="sronek">
        
    </p>
    <article>
        <h2>Wady SPA</h2>
        <p>
            Czas wytworzenia oraz nakład pracy ...
        </p>
    </article>
</section>
<footer>&copy;BP</footer>
</body>
</html>
```

**Listing 7.2. Arkusz style.css**

```
body {
    background:#d4c7b9;
    font-size:18px;
}
h1,h2,footer {
    padding:0.2em;
    text-align:center;
}
.sronek {
    text-align:center;
}
nav{
    float:left;
    width:100%;
}
footer {
    background: #45525a;
    color:#ffe9ba;
}
nav button {
    background: #45525a;
    color:#ffe9ba;
    float:left;
    display:block;
    width:24%;
    margin-right: 1%;
    padding:1% 0;
    font-size:110%;
    text-align:center;
```



Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



Rzeczypospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny

```
border-radius:20px;
}
section {
    clear:both;
}
img {
    max-width: 100%;
    height: auto;
    border-radius:20px;
}
.slajd {
    float:left;
    display:block;
    width:13%;
    text-align:center;
    border-radius:20px;
    background:#ffe9ba;
    padding:1% 0.5%;
    margin:1%;
}
.slajd:hover {
    background:#45525a;
}
footer {
    clear:both;
    border-radius:20px;
}
.galeria{
    margin:auto;
    width:90%;
}
}
@media screen and (max-width: 720px) {
    nav button { width: 48%;
        padding: 10px 0;
        font-size:140%;}
    .slajd {width: 45%; }
}
@media screen and (min-width: 481px) and (max-width: 720px) {
    body { font-size: 13px; }
    h1 { font-size: 23px; }
    h2 { font-size: 18px; }
}
@media screen and (max-width: 481px) {
    body { font-size: 16px; }
    h1 { font-size: 24px; }
    h2 { font-size: 20px; }
    nav button{ font-size:20px;
        padding:10px 0;
        border-bottom: 3px solid white;
        width: 98%;}
    .slajd {width: 95%; }
}
```



**Fundusze Europejskie**  
Wiedza Edukacja Rozwój

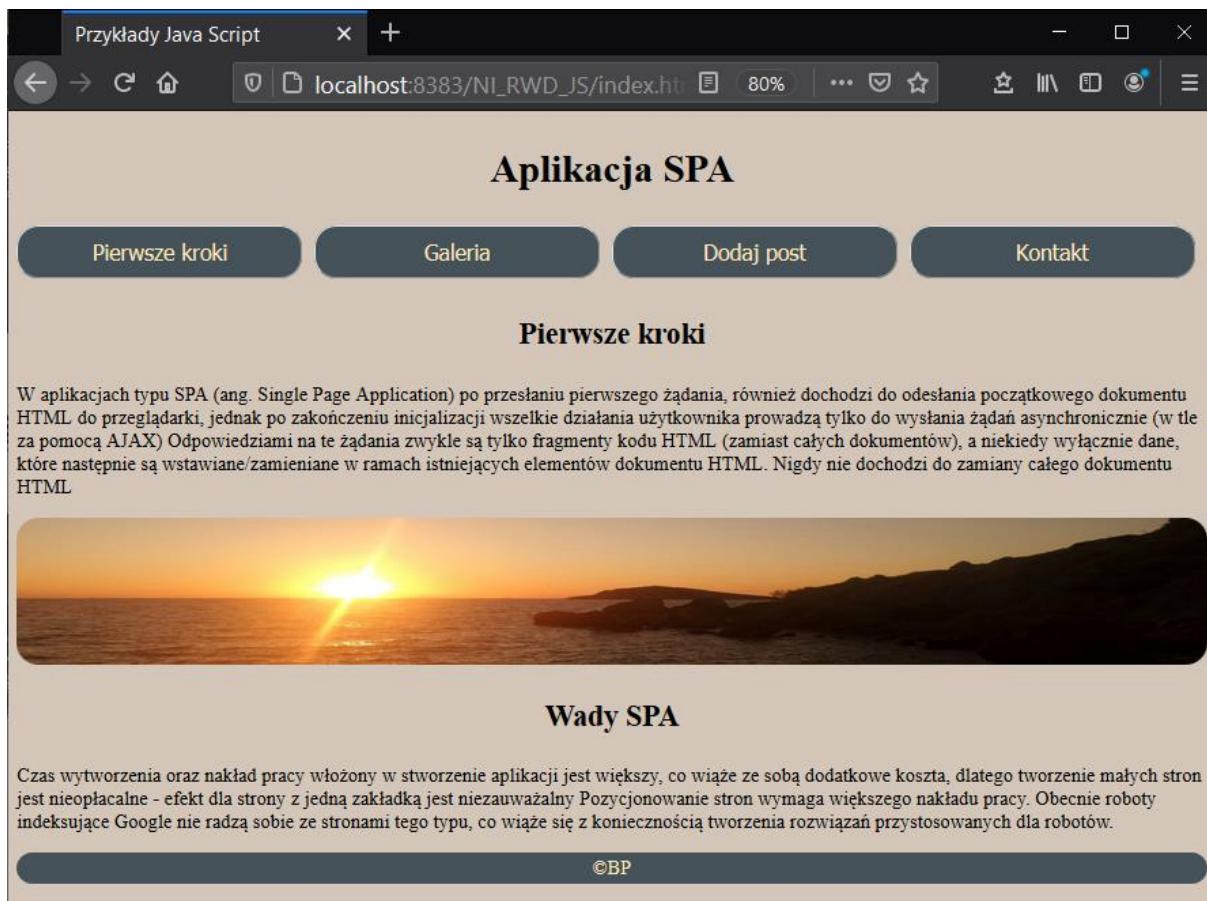


**Rzeczypospolita  
Polska**

**Unia Europejska**  
Europejski Fundusz Społeczny



Rysunek 7.1 prezentuje efekt widoczny w przeglądarce (strona powinna się zachowywać responsywnie).



Rys. 7.1. Początkowy widok strony

Naszym zadaniem będzie teraz dodanie skryptu JS z funkcjami, które umożliwią interakcję z użytkownikiem – w przykładzie będzie to modyfikacja treści sekcji o identyfikatorze ***id="blok"*** z główną zawartością strony.

### Obsługa zdarzeń w JavaScript

Najbardziej podstawowe zdarzenia zachodzące na stronie HTML, związane z akcjami użytkownika, mogą być obsługiwane za pomocą odpowiednich atrybutów dodanych do tagów HTML (w naszym przykładzie będą to przyciski **button**).

Podstawowe zdarzenia zachodzące na stronie to:

- kliknięcie na element – zdarzenie **click**, atrybut w elemencie HTML: **onclick**;
- wejście wskaźnikiem myszki w obszar elementu – **mouseover**, atrybut **onmouseover**;
- wyjście wskaźnika poza obszar elementu – **mouseout**, atrybut **onmouseout**;
- wysłanie formularza realizowane poprzez kliknięcie na przycisk formularza typu **submit** – zdarzenie **submit**, atrybut **onsubmit**;
- wyczyszczenie formularza realizowane poprzez kliknięcie na przycisk formularza typu **reset** – zdarzenie **reset**, atrybut **onreset**.



Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



Rzeczypospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny

### Zadanie 7.2. Funkcje JavaScript do generowania treści strony

W zadaniu 7.1 przygotowany został szablon strony wraz z początkową zawartością sekcji o ***id="blok"***. Naszym zadaniem będzie teraz dodanie odpowiednich funkcji JavaScript, które zostaną wywołane w momencie kliknięcia na poszczególne przyciski. Zwróć uwagę na kod:

```
<button onclick="pokaz(1)">O mnie </button>
```

Dodanie do elementu **button** atrybutu **onclick="pokaz(1)"** powoduje, że w momencie kliknięcia na przycisk (zachodzi zdarzenie **click**) wywołana jest funkcja **pokaz(1)** z jednym parametrem o wartości 1.

Wszystkie definicje tej i innych potrzebnych funkcji umieścimy w pliku *funkcje.js*, który już został dołączony do naszego dokumentu *index.html*. Listing 7.3 przedstawia szablon pliku *funkcje.js*. Zwróć uwagę na instrukcję **switch** oraz na kolejne funkcje, które są wywoływane w zależności od wartości przekazanego funkcji *pokaz* parametru.

Funkcje pomocnicze:

- *function pokaz(id)* – na podstawie wartości parametru *id*, generowana jest zawartość sekcji o *id='blok'*;
- *function pokaz0()* – generuje treść strony początkowej *Pierwsze kroki* (funkcja tworzy wynikowy HTML i zwraca go jako wynik w postaci łańcucha znaków);
- *function pokazGalerie()* – generuje treść strony z galerią;
- *function pokazKontakt()* – generuje treść strony z informacjami kontaktowymi;
- *function pokazPost()* – generuje formularz, za pomocą którego użytkownik może przesyłać post;
- *function pokazDane()* – ma za zadanie zebranie danych z wypełnionego formularza do przesyłania posta i pokazanie odpowiedniego okienka do akceptacji przesyłania danych.

### Listing 7.3. Plik *funkcje.js*

```
function pokaz(id)
{
    var tresp="";
    switch (id)
    {   case 2: tresp += pokazGalerie();break;
        case 3: tresp += pokazPost(); break;
        case 4: tresp += pokazKontakt();break;
        default: tresp += pokaz0();
    }
    //pobierz element o wskazanym id i ustaw jego nową zawartość:
    document.getElementById('blok').innerHTML = tresp;
}

function pokaz0(){
    var tresp = '<h2><br />Pierwsze kroki</h2> ';
    //operator += uzupełnia łańcuch kolejną porcją znaków:
    tresp += '<p> W aplikacjach typu SPA .....</p>'+
        '<p class="srodek"></p>'+
        '<article><h2>Wady SPA</h2><p>'+
        ' Czas wytworzenia oraz nakład pracy ... </p></article>';
    //przekaż wynik – gotową zawartość – do miejsca wywołania funkcji:
    return tresp;
}
```



Fundusze Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny

```
function pokazGalerie()
{
    var tresc='<h2><br />Moja galeria</h2>';
    tresc+= ' <div class="galeria">';
    //wygeneruj kod HTML z obrazami za pomocą pętli for:
    for(i=1;i<=10;i++)
    {
        tresc+='<div class="slajd"> <img ... /></div>';
    }
    return tresc+ '</div>';
}

function pokazKontakt()
{
    var tresc='<h2><br />Kontakt</h2>';
    //zupełnij treść:
    // tresc+...
    return tresc;
}

function pokazPost()
{
    //funkcja generuje kod formularza - dane wpisane w odpowiednie pola przez
    //użytkownika zostaną przekazane mailem na wskazany adres, ale najpierw po
    //zajściu zdarzenia submit (wyślij) - zostanie wywołana funkcja pokazDane()
    tresc='<h2><br />Dodaj post</h2>';
    tresc+='<article class="sródek" ><form action="mailto:b.panczyk@pollub.pl"
        method="post" onsubmit="return pokazDane();">'+
        'Twój email:<br /> <input type="email" name="email" id="email"
        required /><br />'+
        // dodaj kolejne 2 pola formularza
        'Komentarz: <br /><textarea rows="3" cols="20" id="wiadomosc"
        name="wiadomosc" required></textarea>'+
        '<br /> <input type="submit" name="wyslij" value="Wyślij" />'+
        '</form></article>';
    return tresc;
}

function pokazDane()
{
    //Funkcja zbiera dane wpisane w pola formularza i wyświetla okienko
    //typu confirm do zatwierdzenia przez użytkownika:
    var dane="Następujące dane zostaną wysłane:\n";
    dane+="Email: "+document.getElementById('email').value+"\n";
    // uzupełnij dane ...
    if (window.confirm(dane)) return true;
    else return false;
}
```

Rysunek 7.2 przedstawia przykładowy widok strony z galerią.

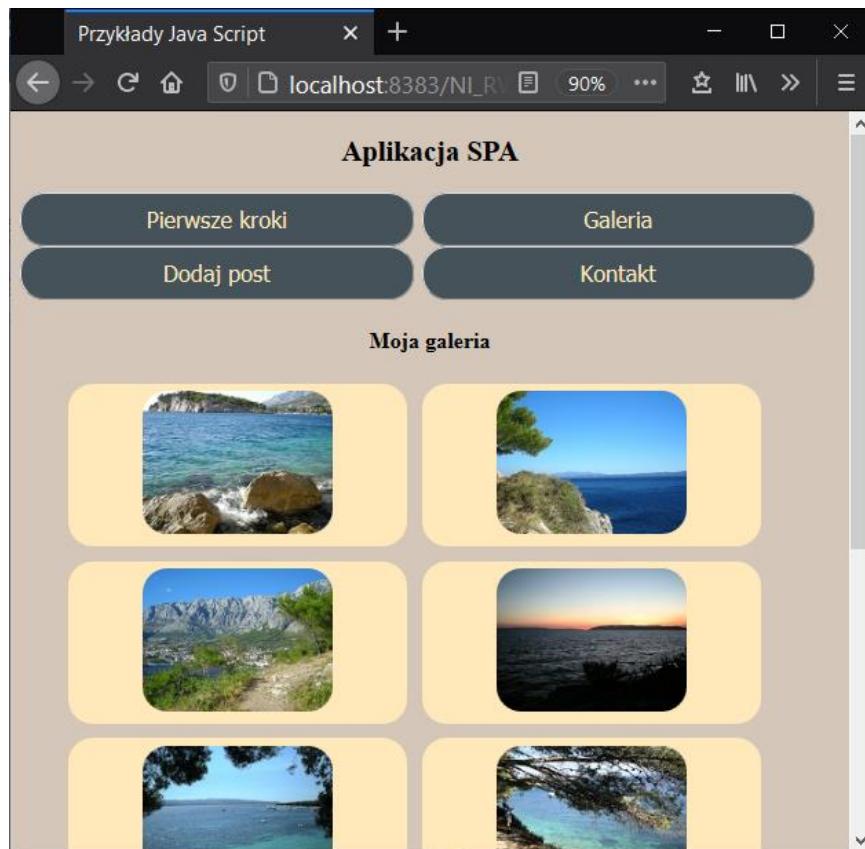


Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska





Rys. 7.2. Widok strony z galerią

### Zadanie 7.3. Dodatkowe elementy w formularzu

Uzupełnij ciało funkcji `pokazPost()` wyświetlającej formularz, tak aby uzyskać efekt widoczny na rysunku 7.3 i przetestuj działanie wszystkich przycisków.

Następnie rozbuduj formularz o:

- pola wyboru na temat zainteresowań (grupa przycisków typu `checkbox` – można zaznaczyć kilka z nich)
- pola wyboru wieku (grupa przycisków typu `radio` – możliwy jest wybór tylko jednej opcji) (Rys. 7.4).

Dodaj też informacje o wyborach użytkownika do wyświetlanego okienka z potwierdzeniem. Przykładowe funkcje sprawdzające, które opcje zostały wybrane przez użytkownika możesz znaleźć w wykładzie z Java Script.



Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



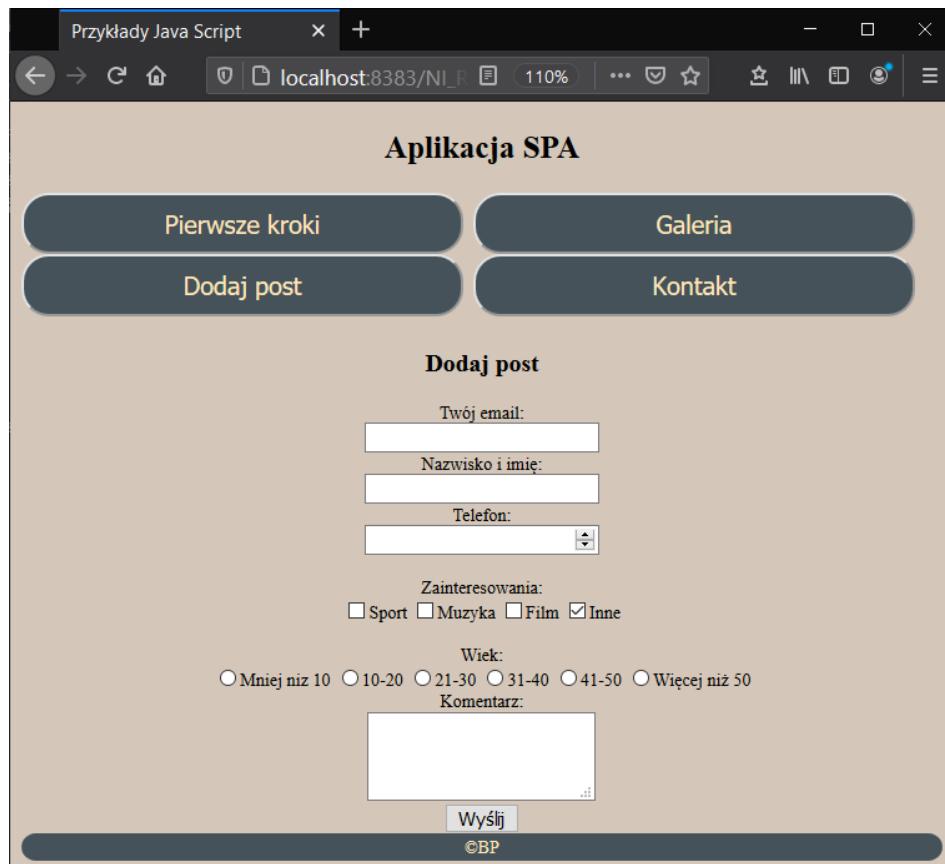
Rzeczpospolita  
Polska

Unia Europejska  
Europejski Fundusz Społeczny





Rys. 7.3. Widok strony z postem



Rys. 7.4. Widok formularza z dodatkowymi polami



Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska

Unia Europejska  
Europejski Fundusz Społeczny



## LABORATORIUM 8. OBSŁUGA I WALIDACJA FORMULARZY PO STRONIE Klienta ZA POMOCĄ JAVASCRIPT.

### Cel laboratorium:

Celem zajęć jest wykorzystanie formularza z projektu realizowanego na laboratoriach 1-3 do przeprowadzenia walidacji za pomocą JavaScript.

### Zakres tematyczny zajęć:

Dodanie niezbędnych do walidacji atrybutów do pól formularza.

Przygotowanie funkcji JavaScript do walidacji.

Wyświetlenie danych zebranych z formularza i przetestowanie walidacji.

### Pytania kontrolne:

1. Jakie obiekty i metody w JavaScript umożliwiają obsługę wyrażeń regularnych?
2. Jaki jest cel walidacji formularzy?
3. Czy walidacja po stronie klienta jest wystarczająca?

### Zadanie 8.1. Walidacja formularza w JavaScript

Walidację danych można realizować po stronie klienta i po stronie serwera. Walidacja po stronie klienta zwykle realizowana jest za pomocą odpowiednich typów pól formularza i ich atrybutów (Lab. 4). Jeszcze większą kontrolę nad danymi możemy uzyskać stosując własne funkcje JavaScript.

Zrealizuj walidację formularza z pliku *formularze.html*. Do walidacji wykorzystaj funkcje pomocnicze umieszczone w zewnętrznym pliku *walidacja.js*.

W pliku powinny się znaleźć funkcje odpowiadające za:

- sprawdzanie poprawności danych wpisanych do pola tekstowego m.in. z wykorzystaniem wyrażeń regularnych,
- sprawdzenie czy i który przycisk typu *radio* został zaznaczony,
- sprawdzenie czy i które przyciski typu *checkbox* zostały wybrane.

Formularz z pliku *formularze.html* uzupełnij o pola, w których zostaną umieszczone ewentualne komunikaty o błędach. Przykładowy kod źródłowy takiego formularza z odpowiednimi komentarzami może mieć postać przedstawioną na Listingu 8.1.

#### **Listing 8.1. Formularz do walidacji uzupełniony komórkami tabeli, w których będą się wyświetlały ewentualne komunikaty o błędach**

```
<form method="post" action="mailto:xxx@xxx.xx" onsubmit="return sprawdz()">
<table>
<tr> <td>Nazwisko: </td>
      <td><input name="nazw" size="30" id="nazw"/> </td>
      <td id="nazw_error" class="czerwone"></td>
</tr>
<tr> <td>Wiek:</td>
      <td><input name="wiek" size = "30" id="wiek"/></td>
      <td id="wiek_error" class="czerwone"></td>
</tr>
```



Fundusze Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny

```

<tr> <td>Państwo:</td>
    <td><select name="kraj" id="kraj">
        <option value="pl" selected="selected">Polska</option>
        <option value="gb">Wielka Brytania</option>
    </select>
</td>
</tr>
<tr> <td>Adres e-mail:</td>
    <td><input name="email" size="30" id="email"/></td>
    <td id="email_error" class="czerwone"></td>
</tr>
</table>

<h4>Zamawiam tutorial z języka:</h4>
<p><input name="php" type="checkbox" id="php"/>PHP
<input name="c" type="checkbox" id="c"/>C/C++
<input name="java" type="checkbox" id="java"/>Java
<span id="produkt_error" class="czerwone"></span></p>

<h4>Sposób zapłaty:</h4>
<p><input name="zaplata" id="zaplata" type="radio" value="euro" />eurocard
<input name="zaplata" type="radio" value="visa" />visa
<input name="zaplata" type="radio" value="przelew" />przelew bankowy
<span id="zaplata_error" class="czerwone"></span><br />
<input type="submit" value="Wyślij" />
<input type="reset" value="Anuluj" /></p>
</form>

```

Zauważ, że w formularzu na **czerwono** oznaczono elementy, w których będą się pojawiały komunikaty o ewentualnych błędach. Każde z pól formularza posiada atrybut ***name***, a te, które będą poddawane walidacji, również atrybut ***id***. Funkcje walidujące wykorzystują metody **getElementById** lub **getElementsByName** do pobrania wartości badanego pola formularza. Walidacja powinna być realizowana za pomocą pomocniczych funkcji umieszczonych w pliku **walidacja.js**, który należy dołączyć do dokumentu HTML znacznikiem **<script>** wewnątrz elementu **<head>**. Całą walidację realizuje funkcja **sprawdz()**, która jest wywoływana w znaczniku **<form>** tuż przed wysłaniem danych na docelowy serwer. Akcja wysłania danych z formularza jest rozpoczynana po kliknięciu na przycisk typu **submit**. Przechwycenie tego zdarzenia umożliwia atrybut **onsubmit** znacznika **form**. W zależności od wyniku zwróconego przez funkcję **sprawdz()**, dane z formularza zostaną przesłane na serwer lub pojawią się stosowne komunikaty o błędach. Pomocnicze funkcje pokazano na Listingu 8.2, a przykładowy efekt walidacji na rys. 8.1.

#### Listing 8.2. Plik walidacja.js

```

function sprawdzPole(pole_id,obiektRegex) {
    //Funkcja sprawdza czy wartość wprowadzona do pola tekstowego
    //pasuje do wzorca zdefiniowanego za pomocą wyrażenia regularnego
    //Parametry funkcji:
    //pole_id - id sprawdzanego pola tekstowego
    //obiektRegex - wyrażenie regularne
    //-----
    var obiektPole = document.getElementById(pole_id);
    if(!obiektRegex.test(obiektPole.value)) return (false);
    else return (true);
}

```



```
function sprawdz_radio(nazwa_radio){  
    //Funkcja sprawdza czy wybrano przycisk radio  
    //z grupy przycisków o nazwie nazwa_radio  
    //-----  
    var obiekt=document.getElementsByName(nazwa_radio);  
    for (i=0;i<obiekt.length;i++)  
    {  wybrany=obiekt[i].checked;  
        if (wybrany) return true;    }  
    return false;  
}  
  
function sprawdz_box(box_id)  
{ //Funkcja sprawdza czy przycisk typu checkbox  
    //o identyfikatorze box_id jest zaznaczony  
    //-----  
    var obiekt=document.getElementById(box_id);  
    if (obiekt.checked) return true;  
    else return false;  
}  
  
function sprawdz()  
{ //Funkcja realizująca sprawdzanie całego formularza  
    //wykorzystując funkcje pomocnicze  
    //-----  
    var ok=true; //zmienna informująca o poprawnym wypełnieniu formularza  
    //Definicje odpowiednich wyrażeń regularnych dla sprawdzenia  
    //poprawności danych wprowadzonych do pól tekstowych  
    obiektNazw = /^[a-zA-Z]{2,20}$/;    //wyrażenie regularne dla nazwiska  
    obiektemail =  
        ^([a-zA-Z0-9])+([.a-zA-Z0-9_-])*@[a-zA-Z0-9_-]+(.[a-zA-Z0-9_-]+)+/;  
    obiektWiek=/^([1-9][0-9]{1,2})$/;  
  
    //Sprawdzanie kolejnych pól formularza.  
    //w przypadku błędu - pojawia się odpowiedni komunikat  
    if (!sprawdzPole("nazw",obiektNazw))  
    {  ok=false;  
        document.getElementById("nazw_error").innerHTML=  
            "Wpisz poprawnie nazwisko!";  
    }  
    else document.getElementById("nazw_error").innerHTML="";  
  
    //Uzupełnij - sprawdź kolejne pola formularza  
    // if ...  
  
    return ok;  
}
```

### Zadanie 8.2. Walidacja i zebranie danych z wypełnionego formularza

Zmodyfikuj funkcję sprawdz() tak, aby po sprawdzeniu danych, ale przed ich wysłaniem pojawiało się okno z informacją jakie dane użytkownik wprowadził do formularza (Rys. 8.1). Aby uniknąć sprawdzania, czy wybrano przycisk z grupy radio – zaznacz za pomocą atrybutu checked="checked" jeden z nich jako wybór domyślny. Będzie potrzebna dodatkowa



Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny

funkcja JavaScript, która sprawdzi, który z przycisków grupy radio został wybrany. Rys. 8.2 pokazuje efekt działania zmodyfikowanej funkcji sprawdz().

**Podstawowe znaczniki formularza HTML to:**

- <form> - znacznik podstawowy zawierający pola formularza
- <input> - znacznik umożliwia wstawienie różnych pól (pole tekstowe, przycisk typu radio, przycisk typu checkbox) w zależności od wartości jego atrybutu type
- <select> - pole formularza typu lista rozwijana
- <textarea> - pole formularza typu obszar tekstowy
- <button> - pole formularza typu przycisk

**Przykładowy formularz HTML:**

Nazwisko:	<input type="text" value="123"/>	Wpisz poprawnie nazwisko!
Wiek:	<input type="text" value="abc"/>	Wpisz poprawnie wiek!
Państwo:	<input type="text" value="Polska"/>	
Adres e-mail:	<input type="text" value="aba.com"/>	Wpisz poprawnie email

**Zamawiam tutorial z języka:**

PHP  C/C++  Java **Musisz wybrać produkt!**

**Sposób zapłaty:**

eurocard  visa  przelew bankowy **Musisz wskazać sposób płatności!**

**Wyślij** **Anuluj**

**Komunikat ze strony localhost:8388**

Dane z wypełnionego przez Ciebie formularza:

Nazwisko: Abacka  
 Wiek: 21  
 Kraj: pl  
 email: aba@pollub.edu.pl  
 Wybrane produkty: C Java .  
 Sposób zapłaty: visa

**OK** **Anuluj**

**Podstawowe znaczniki**

- <form> - znacznik podstawowy zawierający pola formularza
- <input> - znacznik umożliwia wstawienie różnych pól (pole tekstowe, przycisk typu radio, przycisk typu checkbox) w zależności od wartości jego atrybutu type
- <select> - pole formularza typu lista rozwijana
- <textarea> - pole formularza typu obszar tekstowy
- <button> - pole formularza typu przycisk

**Przykładowy formularz HTML:**

Nazwisko:	<input type="text" value="Abacka"/>
Wiek:	<input type="text" value="21"/>
Państwo:	<input type="text" value="Polska"/>
Adres e-mail:	<input type="text" value="aba@pollub.edu.pl"/>

**Zamawiam tutorial z języka:**

PHP  C/C++  Java

**Sposób zapłaty:**

eurocard  visa  przelew bankowy

**Wyślij** **Anuluj**

Rys.8.1. Przykładowe komunikaty o błędach validacji i efekt działania skryptu po poprawnym wprowadzeniu danych



## LABORATORIUM 9. ZASTOSOWANIA BIBLIOTEKI JQUERY W APLIKACJI INTERNETOWEJ

### Cel laboratorium:

Celem zajęć jest poznanie i wykorzystanie możliwości biblioteki jQuery w aplikacji internetowej.

### Zakres tematyczny zajęć:

Dodatek biblioteki do projektu realizowanego na poprzednich laboratoriach.

Zapoznanie się z funkcjami biblioteki.

Wykorzystanie biblioteki w formularzu obliczeniowym.

Wykorzystanie dodatkowych wtyczek do jQuery w celu ciekawszej prezentacji galerii zdjęć.

### Pytania kontrolne:

1. W jaki sposób można dołączyć bibliotekę do dokumentu HTML?
2. Jakie są najważniejsze obiekty jQuery?
3. Wymień najważniejsze akcje, jakie można wykonać na wybranych elementach za pomocą jQuery.

### Zadanie 9.1. Wprowadzenie do jQuery

W celu wykorzystania możliwości biblioteki jQuery należy:

- a) pobrać bibliotekę ze strony projektu <http://jquery.com/> lub użyć biblioteki korzystając z CDN (np. google CDN: <https://developers.google.com/speed/libraries#jquery>) i umieścić w dokumencie HTML, URL do aktualnej wersji biblioteki, np.:

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js">
</script>
```

- b) w dokumencie HTML w znaczniku `<script>` (lub w pliku zewnętrznym z JS) umieścić testowy kod i sprawdzić działanie biblioteki wykonując prostą akcję na znaczniku `<body>`, polegającą na dodaniu do tego elementu tekstu „Test jQuery”:

```
<script>
$(document).ready(function () {
    //akcje do wykonywania po załadowaniu strony
    $('body').text('Test jQuery!');
});
</script>
```

**W ogólnej postaci wywołanie dla wskazanego elementu dokumentu HTML akcji/funkcji w jQuery wygląda następująco:**

```
$( 'element' ).akcja( 'parametry' );
```

gdzie: `$()` – alias do funkcji o nazwie `jQuery()`, tworzącej obiekt `jQuery`;

`element` – wybieranie elementów w oparciu o selektory CSS;

`akcja` – nazwa akcji, która będzie wykonana na wybranych elementach;

`parametry` – parametry dla danej akcji.

Zapoznaj się z tabelami z materiałów wykładowych, gdzie przedstawiono podstawowe elementy i przykłady niezbędne do pracy z biblioteką.

### Przykład zwięzkiego wykorzystania funkcji jQuery:

<pre>var div = \$('#mojDiv');  div.html('&lt;p&gt;jQuery!&lt;/p&gt;'); div.addClass('klasaZcss'); div.css('display', 'none'); div.fadeIn('slow');</pre>	<pre>\$('#mojDiv').html('&lt;p&gt;jQuery!&lt;/p&gt;')     .addClass('klasaZcss')     .css('display', 'none')     .fadeIn('slow');</pre>
---	---

### Zadanie 9.2. Podstawy pracy z jQuery

W dokumencie *obliczenia.html* do wyznaczania rat spłaty kredytu, wykorzystaj możliwości dostępu do elementów za pomocą krótszego zapisu w jQuery.

Na początek przetestuj działanie podstawowych metod:

- ustal szary kolor tła dla elementu `#tresc`,
- dla wszystkich pól tekstowych ustaw pogrubienie czcionki,
- zmień kolor tła pól tekstowych z wynikami obliczeń na jasnozielony (dodaj do nich klasę np. `zielony`, której definicję należy dopisać do reguł CSS),
- na koniec obsłuż akcję kliknięcia przycisku „Oblicz” (wykorzystaj podobny przykład z wykładu).

### Zadanie 9.3. Galeria z wykorzystaniem jQuery lightbox

Pobierz dodatkową wtyczkę do jQuery ze strony:

<https://lokeshdhakar.com/projects/lightbox2/>.

Po rozpakowaniu, w folderze `src` znajdziesz m.in. foldery: `css`, `js` i `images`. Folder `images` z całą zawartością dodaj jako kolejny folder do swojego projektu, a pliki z folderów `css` (`lightbox.css`) i `js` (`lightbox.js`) skopiuj do odpowiednich folderów już istniejących w projekcie. Zastosuj `ligthbox` do strony *galeria.html*.

W tym celu:

- w nagłówku strony dołącz **najpierw** bibliotekę `jQuery` (jak w zadaniu 9.1), dodatkowo wtyczkę `lightbox.js` oraz podepnij arkusz `lightbox.css`:  

```
<script src=
"https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js">
</script>
<script src="js/lightbox.js"></script>
<link rel="stylesheet" href="css/lightbox.css" />
```
- w kodzie HTML zmodyfikuj odpowiednie hiperłącza do zdjęć tak, aby włączyć `lightbox` - dodaj atrybut **`data-lightbox`** o takiej samej wartości **dla wszystkich** linków do obrazów wchodzących w skład galerii. Na przykład:  

```
<a href="zdjecia/obraz1.JPG" data-lightbox="galeria">
```

Można też dodatkowo dodać opis dla każdego zdjęcia za pomocą kolejnego atrybutu **`data-title`**, np.  

```
<a href="zdjecia/obraz1.JPG" data-lightbox="galeria"
data-title="Obraz 1">
```



Fundusze Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska



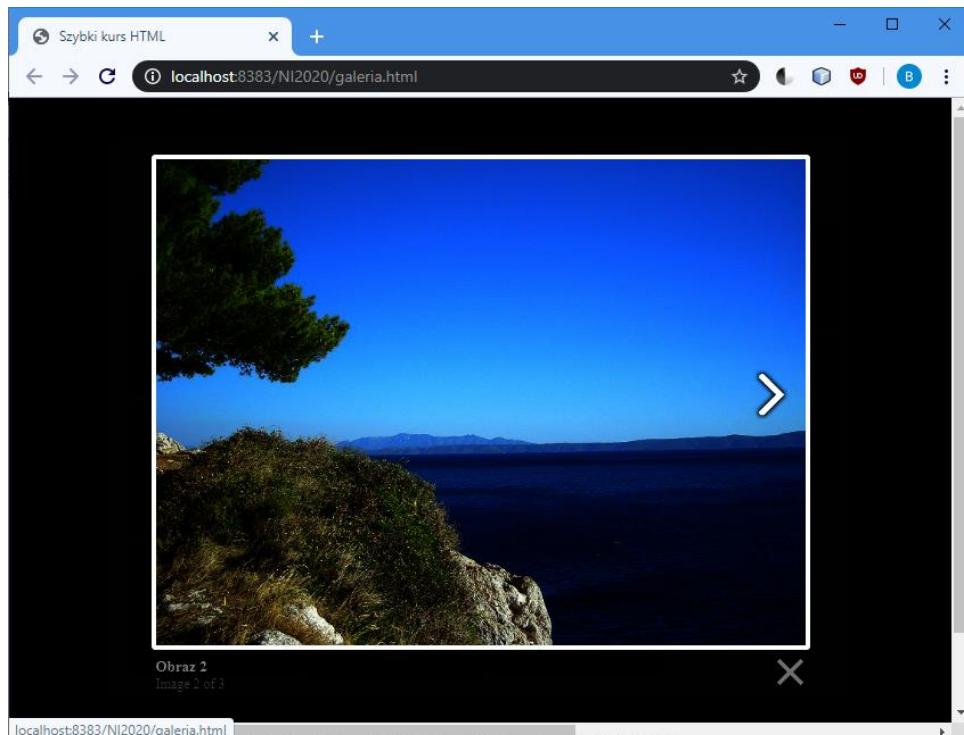
Unia Europejska  
Europejski Fundusz Społeczny

Punkty *a)* i *b)* wystarczą już do tego, aby nasza galeria przyjęła postać jak na rys. 9.1, Odwołania z miniatur do zdjęć pozostają bez zmian i są postaci np.:

```
<a href="zdjecia/obraz1.JPG" data-lightbox="galeria" data-title="Obraz 1">
    
</a>
```

Wykorzystując bibliotekę **jQuery** można tworzyć różne atrakcyjne galerie. Przykłady z tutorialami możesz znaleźć np. na stronie:

<http://vandelaydesign.com/blog/web-development/jquery-image-galleries/>



Rys 9.1. Galeria z wykorzystaniem jQuery lightbox

#### Zadanie 9.4. Prosty slider z jQuery

- Utwórz nową stronę *slide.html* (<http://jonraasch.com/blog/a-simple-jquery-slideshow>) (Listing 9.1).

#### Listing 9.1. Plik *slide.html*

```
<!DOCTYPE html>
<html>
    <head>
        <title>Slide show</title>
        <meta charset="UTF-8">
        <link rel="stylesheet" type="text/css" href="CSS/slide.css" />
        <script src=
            "https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js">
        </script>
        <script type="text/javascript" src="JS/slide.js"></script>
    </head>
    <body>
        <div id="content">
```



Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny

```
<h2>Prosty slider w jQuery</h2>
<div id="slideshow">
    
    
    
</div>
</div>
</body>
</html>
```

- b) Następnie, korzystając z właściwości ***z-index*** w CSS, aktywny obraz możemy pokazać przed pozostałymi („na wierzchu”) (*slide.css* - Listing 9.2).

**Listing 9.2. Plik slide.css**

```
#slideshow img{
    width:100%;
    height:200px;
}
#slideshow {
    position:relative;
    height:200px;
}
#slideshow img {
    position:absolute;
    top:0;
    left:0;
    z-index:8;
}
#slideshow img.active {
    z-index:10;
}
#slideshow img.last-active {
    z-index:9;
}
```

- c) W pliku *slide.css* dla obrazów zawartych w elemencie *div* o *id="slideshow"* zdefiniowano ich wymiary: szerokość 100% i wysokość 200px. Kolejne reguły definiują trzy różne wartości dla współrzędnej ***z-index*** - manipulując tymi współrzędnymi za pomocą jQuery osiągniemy efekt animacji kolejnych obrazów – „*wyciągamy na wierzch*” obraz aktywny. Realizuje to kod *slide.js* (Listing 9.3).

**Listing 9.3. Plik slide.js**

```
function slideSwitch() {
    var $active = $('#slideshow IMG.active');
    if ($active.length === 0)
        $active = $('#slideshow IMG:last');
    var $next = $active.next().length ? $active.next()
        : $('#slideshow IMG:first');
    $active.addClass('last-active');

    $next.css({opacity: 0.0})
        .addClass('active')
        .animate({opacity: 1.0}, 1000, function () {
            $active.removeClass('active last-active');
```



Fundusze Europejskie  
Wiedza Edukacja Rozwój

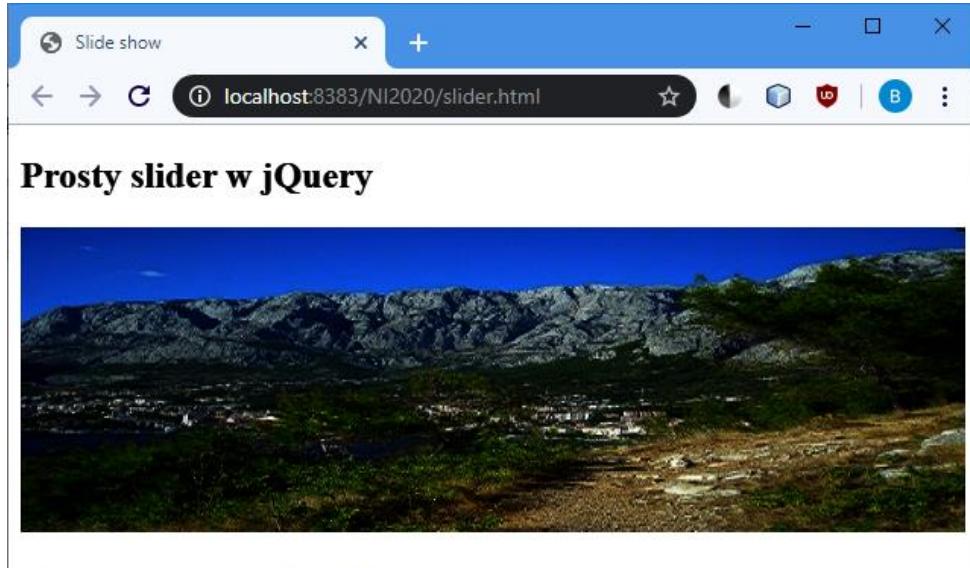


Rzeczypospolita  
Polska



```
});  
}  
$(function () {  
    setInterval("slideSwitch()", 3000);  
});
```

Wynik końcowy przedstawia rysunek 9.2.

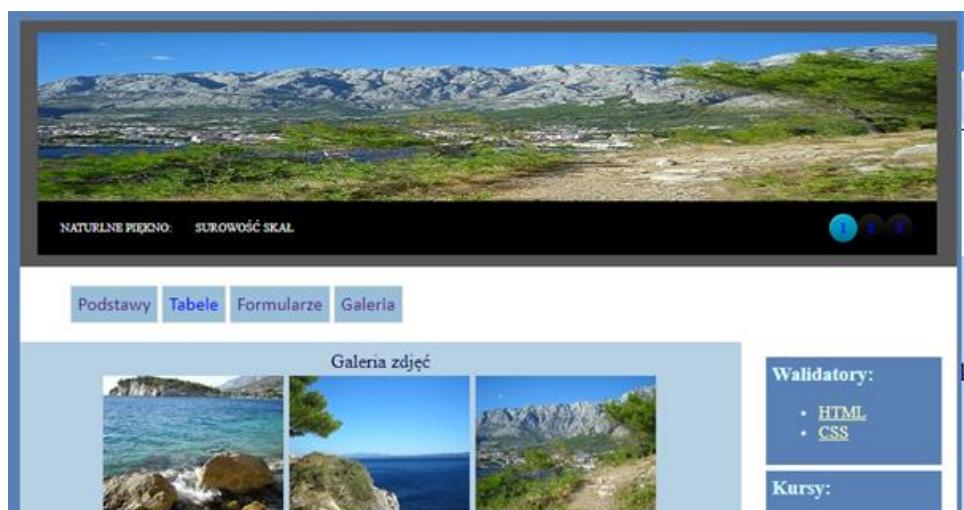


Rys.9.2. Przykładowy efekt działania slidera

### Zadanie 9.5. Animacja zdjęć w projekcie

Wykorzystaj dowolną animację zdjęć do swojego projektu tak, aby uzyskać efekt animacji obrazów w banerze, podobnie jak na rysunku 9.3.

Darmowe, responsywne animacje zdjęć możesz znaleźć np. na stronie: <https://freshdesignweb.com/jquery-image-slider-slideshow/>



Rys.9.3. Zmodyfikowany slider w banerze na stronie z galerią



Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny

## LABORATORIUM 10. TWORZENIE RESPONSYWNEGO SZABLONU STRONY Z ZASTOSOWANIEM SZKIELETU BOOTSTRAP

### Cel laboratorium:

Celem zajęć jest poznanie i wykorzystanie możliwości framework'a Bootstrap.

### Zakres tematyczny zajęć:

Dołączenie biblioteki klas CSS i funkcji JavaScript udostępnianych przez Bootstrap.  
Wykorzystanie szablonu Bootstrap do utworzenia projektu.  
Zapoznanie się z klasami CSS dostępnymi we framework'u.  
Zastosowanie w projekcie rozwiązań typu responsywnej nawigacji, układy kolumnowe, responsywny slider itp.

### Pytania kontrolne:

1. Co oferuje Bootstrap i dlaczego warto z niego korzystać?
2. W jaki sposób można zbudować responsywną siatkę w Bootstrap?
3. Jakie są najważniejsze klasy CSS dostępne w Bootstrap?

### Zadanie 10.1. Adaptacja wybranego szablonu Bootstrap do potrzeb swojego projektu

1. Otwórz stronę <https://startbootstrap.com/>.
2. Pobierz szablon *Small Business* z dostępnych tam, darmowych szablonów (**Templates**) (Rys. 10.1).
3. Rozpakuj i przejrzyj pobrane zasoby. Zapoznaj się z warunkami licencji MIT.
4. Utwórz swój nowy projekt i skopiuj tam plik *index.html*, folder *css* (z plikiem *small-business.css*) oraz folder *vendor* z zawartością. Przejrzyj kod źródłowy pliku *index.html* i *small-business.css*. Zwróć uwagę (w pliku *index.html*) na elementy *<link>* oraz na bloki *<script>*.
5. Otwórz plik *index.html* w przeglądarce i za pomocą narzędzi deweloperskich sprawdź ile żądań (i o jakie zasoby) zostało wysłanych w celu pobrania zawartości (zdarzenie **load** obiektu **document**) tej jednej strony.
6. Sprawdź responsywność szablonu.

### Zadanie 10.2. Modyfikacja szablonu

Zmodyfikuj plik *index.html*, tak aby uzyskać efekt podobny do tych pokazanych na rysunku 10.2 i 10.3.

W tym celu:

1. Zmień opcje w menu i ustaw linki na odpowiednie strony (**Politechnika Lubelska, Katedra Informatyki, Moodle**).
2. W miejsce banera wstaw obraz pobrany ze strony (lub inny): [https://cdn.pixabay.com/photo/2018/02/15/18/29/devops-3155972\\_340.jpg](https://cdn.pixabay.com/photo/2018/02/15/18/29/devops-3155972_340.jpg)
3. Zmodyfikuj treści odpowiednich elementów HTML i ustaw linki do tutoriali w3school (Rys. 10.3).
4. Zmień treść w stopce.
5. Możesz również spróbować zmodyfikować np. kolorystykę projektu.



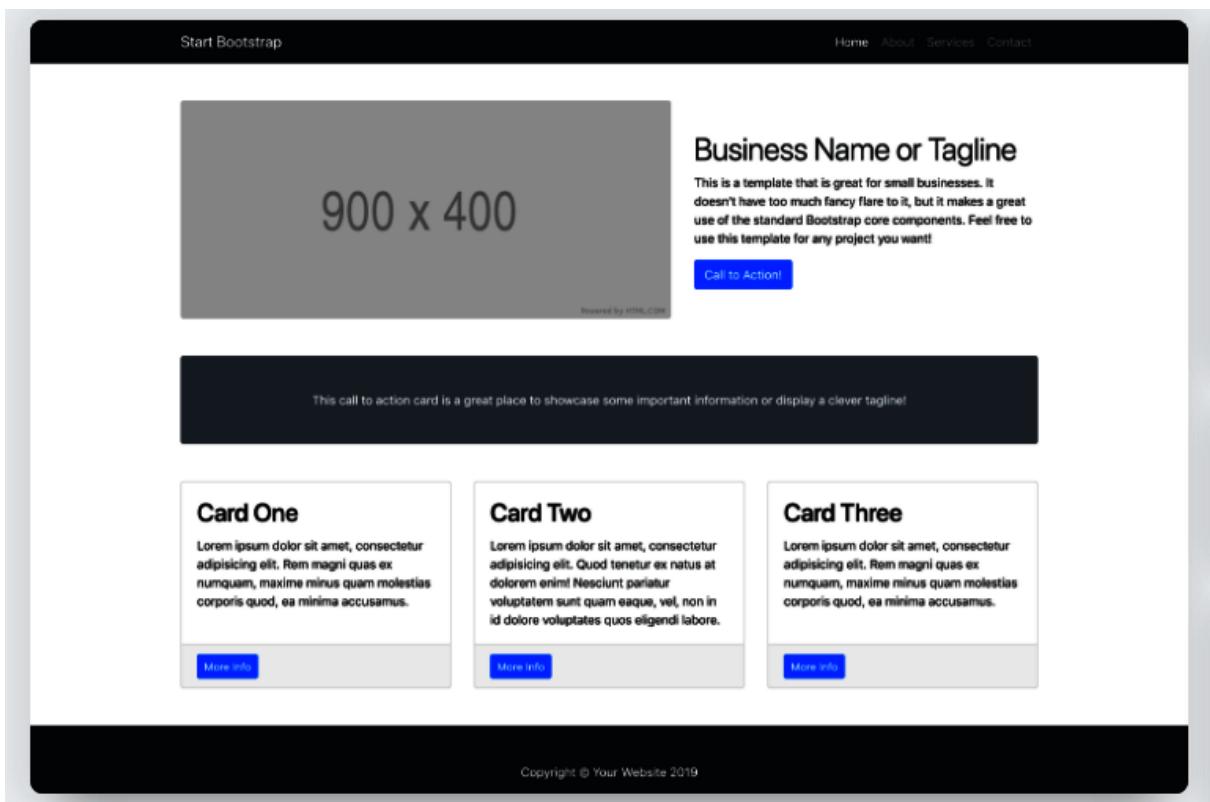
Fundusze Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska



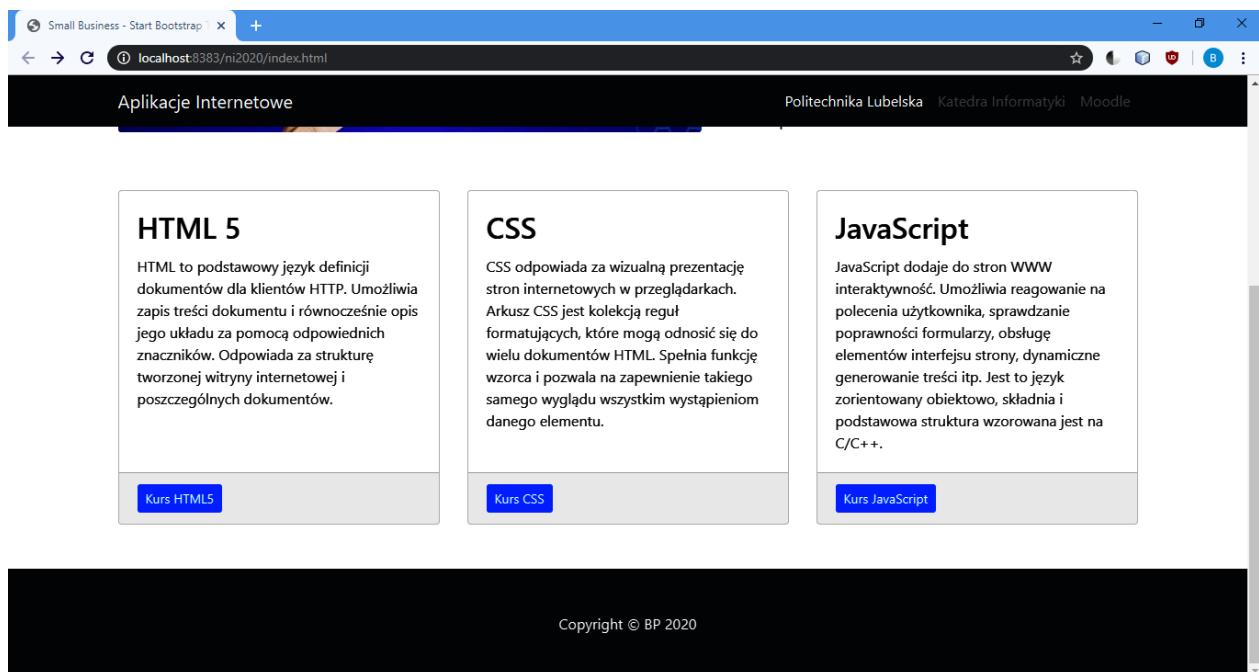
Unia Europejska  
Europejski Fundusz Społeczny



Rys. 10.1. Small business template

A screenshot of a modified template titled "Small Business - Start Bootstrap". The header shows the URL "localhost:8383/ni2020/index.html". The main content area features a large circular diagram with the words "PLAN", "BUILD", "CONTINUOUS", "INTEGRATION", "DEPLOY", and "OPERATE" around its perimeter, with a hand pointing at it. To the right of the diagram is a section titled "Podstawy aplikacji internetowych" with text about web application components and models. Below this are three cards: "HTML 5" (describing HTML as a document definition language), "CSS" (describing CSS for visual presentation), and "JavaScript" (describing JavaScript for interactivity). Each card has a link to "https://moodle.cs.pollub.pl".

Rys. 10.2. Modyfikacje szablonu



Rys. 10.3. Modyfikacje szablonu

### Zadanie 10.3. Bootstrap carousel

Korzystając z kodu np. na stronie <https://bs4.kursbootstrap.pl/slider-carousel/> (sprawdź, co oznaczają klasy Bootstrapa dodane do elementów *div* i *img*), zamiast pojedynczego zdjęcia w banerze - dodaj animację 3 obrazów (Rys. 10.4). Dodatkowe obrazy pobierz np. ze strony: <https://pixabay.com/pl/illustrations/search/?cat=computer>.

Przykładowe obrazy:

[https://cdn.pixabay.com/photo/2020/02/15/14/19/network-4851079\\_340.jpg](https://cdn.pixabay.com/photo/2020/02/15/14/19/network-4851079_340.jpg)

[https://cdn.pixabay.com/photo/2020/02/15/16/56/technology-4851446\\_340.jpg](https://cdn.pixabay.com/photo/2020/02/15/16/56/technology-4851446_340.jpg)

[https://cdn.pixabay.com/photo/2020/01/24/06/46/ball-4789466\\_340.jpg](https://cdn.pixabay.com/photo/2020/01/24/06/46/ball-4789466_340.jpg)

Aby efekt animacji był zadowalający, obrazy powinny mieć **takie same** wymiary (w szczególności wysokość). Na potrzeby przykładu, ustal wysokość obrazów, dodając do elementów *img* w karuzeli atrybut: *style="height: 350px"* .



Fundusze  
Europejskie  
Wiedza Edukacja Rozwój

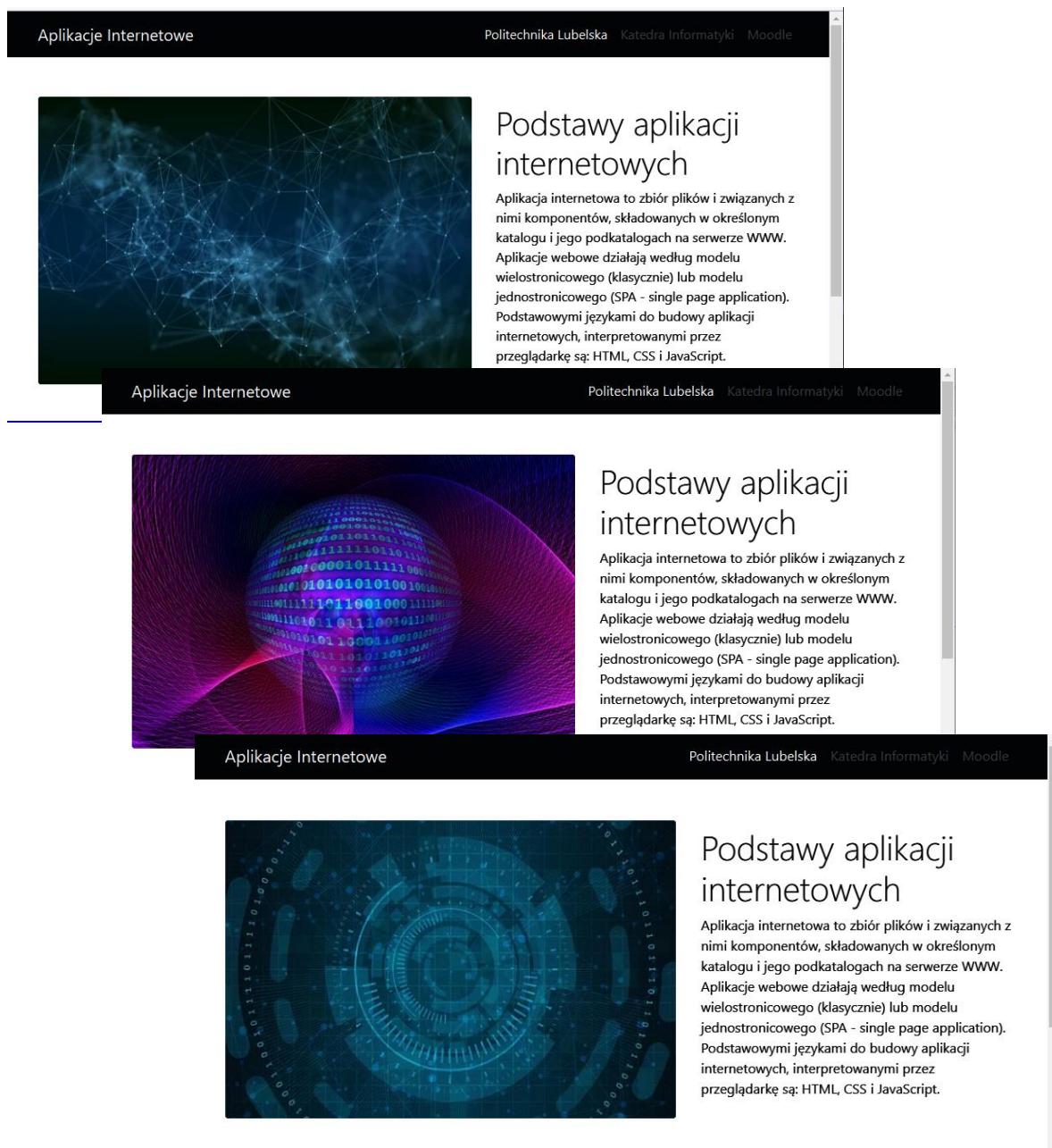


Rzeczpospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny

## Zintegrowany Program Rozwoju Politechniki Lubelskiej – część druga



Rys. 10.4. Karuzela Bootstrap

### Zadanie 10.4. Inne możliwości Bootstrapa

W pliku *index.html* dodaj czwartą kartę w części:

```
<!-- Content Row --> <div class="row"> ... </div>
```

tak, aby zachować responsywność i przy zmniejszaniu szerokości okna przeglądarki, układ kart zmieniał się od 4x1 stopniowo na 2x2 i 1x4 (Rys. 10.5).

Skorzystaj z przykładów podanych na stronach, np.:

<https://bs4.kursbootstrap.pl/>

<https://www.w3schools.com/bootstrap4/default.asp>

i wykorzystaj inne rozwiązania korzystając z odpowiednich klas Bootstrapa.



Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny

## Zintegrowany Program Rozwoju Politechniki Lubelskiej – część druga

Aplikacje Internetowe

Politechnika Lubelska Katedra Informatyki Moodle

<b>HTML 5</b> <p>HTML to podstawowy język definicji dokumentów dla klientów HTTP. Umożliwia zapis treści dokumentu i równocześnie opis jego układu za pomocą odpowiednich znaczników. Odpowiada za strukturę tworzonej witryny internetowej i poszczególnych dokumentów.</p> <p><a href="#">Kurs HTML5</a></p>	<b>CSS</b> <p>CSS odpowiada za wizualną prezentację stron internetowych w przeglądarkach. Arkusz CSS jest kolekcją reguł formatujących, które mogą odnosić się do wielu dokumentów HTML. Spełnia funkcję wzorca i pozwala na zapewnienie takiego samego wyglądu wszystkim wystąpieniom danego elementu.</p> <p><a href="#">Kurs CSS</a></p>	<b>JavaScript</b> <p>JavaScript dodaje do stron WWW interaktywność. Umożliwia reagowanie na polecenia użytkownika, sprawdzanie poprawności formularzy, obsługę elementów interfejsu strony, dynamiczne generowanie treści itp. Jest to język zorientowany obiektowo, składnia i podstawowa struktura wzorowana jest na C/C++.</p> <p><a href="#">Kurs JavaScript</a></p>	<b>jQuery</b> <p>Query – lekka biblioteka programistyczna dla języka JavaScript, ułatwiająca korzystanie z JavaScriptem (w tym manipulację drzewem DOM). Pozwala osiągnąć interesujące efekty animacji, dodać dynamiczne zmiany strony, wykonać zapytania AJAX.</p> <p><a href="#">Kurs jQuery</a></p>
--	---	--	--

Aplikacje Internetowe

<b>HTML 5</b> <p>HTML to podstawowy język definicji dokumentów dla klientów HTTP. Umożliwia zapis treści dokumentu i równocześnie opis jego układu za pomocą odpowiednich znaczników. Odpowiada za strukturę tworzonej witryny internetowej i poszczególnych dokumentów.</p> <p><a href="#">Kurs HTML5</a></p>	<b>CSS</b> <p>CSS odpowiada za wizualną prezentację stron internetowych w przeglądarkach. Arkusz CSS jest kolekcją reguł formatujących, które mogą odnosić się do wielu dokumentów HTML. Spełnia funkcję wzorca i pozwala na zapewnienie takiego samego wyglądu wszystkim wystąpieniom danego elementu.</p> <p><a href="#">Kurs CSS</a></p>
<b>JavaScript</b> <p>JavaScript dodaje do stron WWW interaktywność. Umożliwia reagowanie na</p>	<b>jQuery</b> <p>Query – lekka biblioteka programistyczna dla języka JavaScript, ułatwiająca</p>

Rys. 10.5. Responsywny układ kart



**Fundusze Europejskie**  
Wiedza Edukacja Rozwój



**Rzeczypospolita  
Polska**

**Unia Europejska**  
Europejski Fundusz Społeczny



## LABORATORIUM 11. CZEŚĆ 1. BUDOWA APLIKACJI WYKORZYSTUJĄCEJ LOKALNE MAGAZYNY DANYCH

### Cel laboratorium:

Celem zajęć jest poznanie i zastosowanie w aplikacjach możliwości utrwalania danych za pomocą *Web Storage API*.

### Zakres tematyczny zajęć:

Budowa aplikacji korzystającej z lokalnych magazynów danych: *localStorage* i *sessionStorage*.

Zastosowanie formatu JSON do zapisu danych w lokalnych magazynach.

### Pytania kontrolne:

1. Czym różni się *localStorage* od *sessionStorage*?
2. Jakie są podstawowe metody do pracy z *sessionStorage* i *localStorage*?
3. Podaj charakterystykę formatu JSON.
4. Jakie metody obiektu JSON pozwalają na konwersję obiektu JavaScript do łańcucha i odwrotnie?

### Zadanie 11.1. Praca z obiektem *sessionStorage*

Stwórz formularz zawierający dwa pola tekstowe do wprowadzenia kodu szesnastkowego i nazwy koloru: **RGB** (6 cyfr szesnastkowych) oraz **nazwa**. Do formularza dodaj trzy przyciski opisane jako **Zapisz parę**, **Pokaż wszystkie pary** i **Usuń dane** (Rys. 11.1). Akcja kliknięcia na te przyciski ma wywoływać odpowiednie funkcje JavaScript do:

- zapisu danych do *sessionStorage*,
- odczytu wszystkich danych zapisanych w *sessionStorage* i wyświetlenia ich pod formularzem,
- usunięcia wszystkich wprowadzonych danych z magazynu sieciowego *sessionStorage*.

Pamiętaj o sprawdzeniu, czy przeglądarka obsługuje *sessionStorage*.

Wykorzystaj metody obiektu *sessionStorage*:

- **getItem(key)** – zwraca wartość dla danego klucza lub **null**, jeśli klucz nie istnieje,
- **setItem(key, value)** – zapisuje dane w postaci klucz-wartość,
- **removeItem(key)** – usuwa element o podanym kluczze,
- **key(position)** – zwraca klucz dla wartości umieszczonej pod określoną pozycją,
- **clear()** – usuwa wszystkie pary klucz-wartość.

Do pobrania wszystkich danych z *sessionStorage* wykorzystaj np. pętlę:

```
for(var i=0;i<sessionStorage.length;i++) { }
```



Fundusze Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny

Zapisane pary klucz-wartość możesz sprawdzić w przeglądarce *Chrome* (rys. 11.1), korzystając z zakładki ***Application***, dostępnej w narzędziach deweloperskich. Sprawdź działanie strony dla dwóch okien przeglądarki otwartych jednocześnie. Co się dzieje po zamknięciu przeglądarki – czy po jej ponownym otwarciu dane w ***sessionStorage*** są dostępne?

The screenshot shows a browser window with the title "Web storage" and the URL "localhost:8383/NI\_2020/sessionStoragetest.html". The page content includes a form with fields for color code and name, and buttons for saving, viewing all pairs, and deleting data. Below the form, there are four colored boxes: Green, Red, Blue, and White, each with its corresponding color code. The developer tools are open, specifically the "Application" tab under the "Storage" section. The "Session Storage" table shows the following data:

Key	Value
00FF00	Green
0000FF	Blue
FFFFFF	White
FF0000	Red

A red box highlights the "Session Storage" section in the sidebar, and another red box highlights the "Application" tab in the top bar.

Rys. 11.1. Przykładowy efekt działania strony

### Zadanie 11.2. Praca z obiektem ***localStorage***

Zrealizuj to samo zadanie, ale korzystając z obiektu ***localStorage***. Sprawdź ponownie działanie strony dla dwóch okien przeglądarki otwartych jednocześnie. Co się dzieje po zamknięciu przeglądarki – czy po jej ponownym otwarciu dane w ***localStorage*** są dostępne? Czy widać różnicę pomiędzy ***localStorage*** a ***sessionStorage***?

### Zadanie 11.3. Zastosowanie formatu JSON – aplikacja koszyka na produkty

1. Utwórz stronę HTML zawierającą formularz z polami:

- nazwa produktu,
- cena,
- kolor,
- liczba sztuk.



Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny

2. Do formularza dodaj trzy przyciski wraz z kodem JS do ich obsługi:

- „*Zapisz produkt do koszyka*” - zapisuje dane z formularza do *localStorage*;
- „*Wyświetl koszyk*” – wyświetla listę wszystkich produktów dodanych do koszyka na stronie w **postaci tabeli**, a w przypadku braku produktów wyświetlany jest stosowny komunikat;
- „*Usuń wszystkie produkty*” – usuwa wszystkie elementy zapisane w *localStorage*.

#### **UWAGA**

Ponieważ formularz złożony jest z większej liczby danych – wartości pobrane z pól formularza wykorzystaj do utworzenia obiektu JSON i taki obiekt zapisz do magazynu *localStorage* (listing 11.1).

#### **Listing 11.1. Zapis obiektu do localStorage**

```
//utwórz obiekt o odpowiednich atrybutach:  
var item = {};  
item.cena = 200;  
item.nazwa = "Spodnie";  
//dodaj kolejne właściwości do obiektu item:  
item...  
//zapisz obiekt do localStorage - wykorzystaj metodę stringify klasy JSON  
localStorage.setItem('item_1', JSON.stringify(item));  
//odczytaj obiekt z localStorage - wykorzystaj metodę parse klasy JSON  
var retrieveItem = JSON.parse(localStorage.getItem('item'));
```

3. Sprawdź właściwości obiektu, korzystając z konsoli przeglądarki za pomocą polecenia `console.log(item)`.
4. Sprawdź działanie strony dla dwóch okien przeglądarki otwartych jednocześnie. Czy dwa odrębne okna mają dostęp do tego samego magazynu *localStorage*?
5. Zmodyfikuj metodę dodawania produktów do *localStorage* – wykorzystaj możliwość zapisu listy obiektów pod jednym kluczem (wykorzystaj przykład **Organizer** z wykładu).

#### **Zadanie 11.4. Edycja i usuwanie danych z koszyka**

Zmodyfikuj stronę i skrypty utworzone w zadaniu 11.3, tak aby była możliwa edycja i usuwanie pojedynczych elementów z koszyka. Spróbuj też dodać wyszukiwarkę np. po nazwie produktu (lub jej części – zastosuj odpowiednie wyrażenie regularne).



Fundusze Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny

## LABORATORIUM 11. CZEŚĆ 2. BUDOWA APLIKACJI WYKORZYSTUJĄCEJ GEOLOKALIZACJĘ.

### Cel laboratorium:

Celem zajęć jest poznanie i zastosowanie w aplikacjach możliwości **Geolocation API**.

### Zakres tematyczny zajęć:

Zastosowanie mechanizmu geolokalizacji z wykorzystaniem **Google Maps API**.

### Pytania kontrolne:

1. W jaki sposób można uzyskać informacje o współrzędnych geograficznych położenia urządzenia za pomocą API HTML5 i JavaScript?
2. Podaj właściwości i metody obiektu **Position**.

### Zadanie 11.5. Praca z obiektem Position

Sprawdź działanie kodu przedstawionego na listingu 11.2. Następnie zmodyfikuj go tak, aby wyświetlał nie tylko szerokość, ale również długość geograficzną, na której się znajdujesz (Rys. 11.2).

#### **Listing 11.2. Przykładowy kod strony HTML wykorzystującej geolokalizację**

```
<!DOCTYPE html>
<html>
    <head>
        <title>Geolokalizacja</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <script>
            function showLocation(position) {
                var latitude = position.coords.latitude;
                var output = document.getElementById("geo");
                output.innerHTML = "<p>Szerokość geograficzna: " + latitude +
                    "</p>";
            }
            function errorHandler(error) {
                var output = document.getElementById("geo");
                switch (error.code) {
                    case error.PERMISSION_DENIED:
                        output.innerHTML = "Użytkownik nie udostępnił danych.";
                        break;
                    case error.POSITION_UNAVAILABLE:
                        output.innerHTML = "Dane lokalizacyjne niedostępne.";
                        break;
                    case error.TIMEOUT:
                        output.innerHTML = "Przekroczono czas żądania.";
                        break;
                    case error.UNKNOWN_ERROR:
                        output.innerHTML = "Wystąpił nieznany błąd.";
                }
            }
        </script>
    </head>
    <body>
        <div id="geo"></div>
    </body>
</html>
```



Fundusze Europejskie  
Wiedza Edukacja Rozwój



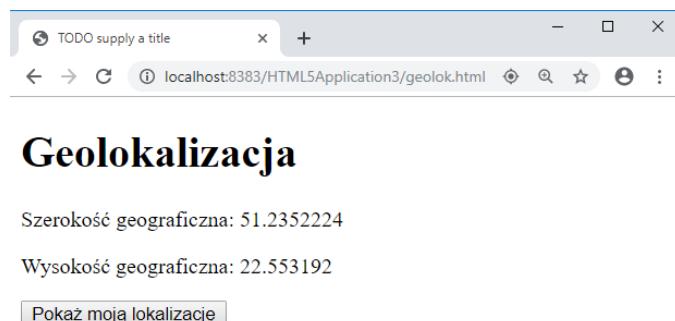
Rzeczpospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny

```
        break;
    }
}
function getLocation() {
    if (navigator.geolocation) {
        var options = {timeout: 60000};
        navigator.geolocation.getCurrentPosition(
            showLocation,
            errorHandler,
            options);
    } else { alert("Twoja przeglądarka nie wspiera geolokalizacji!");}
}
</script>
</head>
<body>
    <h1>Geolokalizacja</h1>
    <div id="geo"></div>
    <p><button onclick="getLocation()">Pokaż moją lokalizację</button></p>
</body>
</html>
```

W przypadku trudności - skorzystaj z konsoli dostępnej w przeglądarce w celu znalezienia błędu działania skryptu JS.



Rys. 11.2. Strona ze współrzędnymi położenia

### Zadanie 11.6. Praca z Google Maps API

Wykorzystując skrypt przygotowany w zadaniu 11.5, rozszerz stronę HTML o statyczną mapę (rys. 11.3) w oparciu o API Google Maps. Od wersji Google Maps API 3 wymagany jest klucz (<https://developers.google.com/maps/documentation/javascript/get-api-key>), który można uzyskać z chmury Google (Google Cloud Platform): <https://cloud.google.com/maps-platform>. Można też wypróbować bezpłatnie Google Cloud Platform: <https://console.cloud.google.com/freetrial/signup/tos>.

Każdy deweloper powinien uzyskać swój własny klucz. W naszym przykładzie pokazane będzie jak umieścić mapkę na stronie, ale nie będziemy pobierać klucza. Bez klucza – pojawi się komunikat o braku możliwości wyświetlenia mapy w prawidłowym widoku i uzyskamy obraz jak na rysunku 11.3.



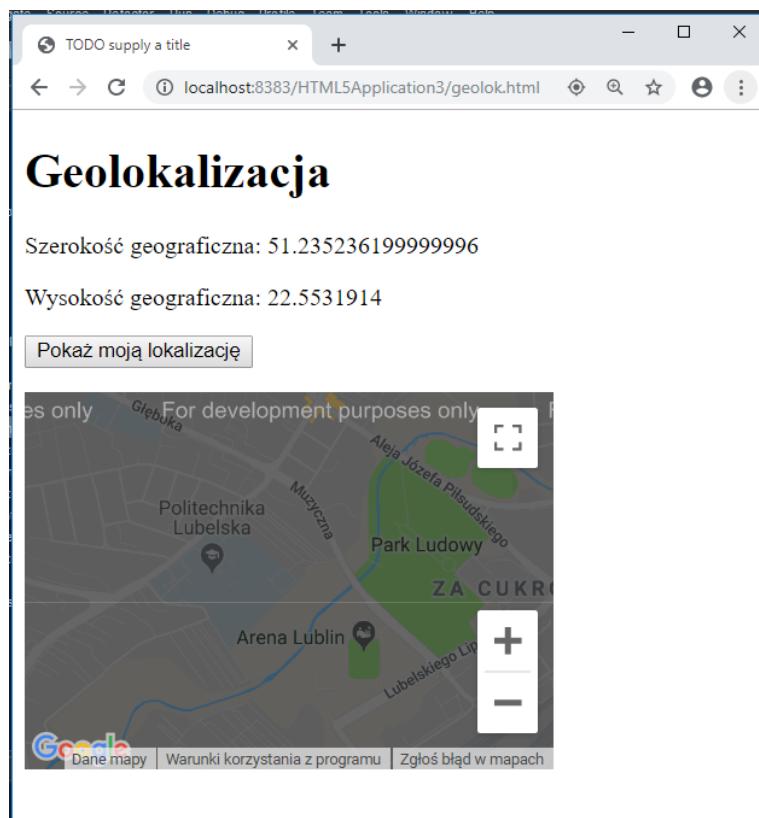
Fundusze Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny



Rys. 11.3. Strona z mapą

W celu osadzenia mapki na stronie HTML należy:

- ```
<script src="https://maps.google.com/maps/api/js?sensor=false" >
</script>
• wstawić obiekt mapy do elementu <div> w HTML:
<div id="mapka" style="width:350px; height:250px;">
    <!-- tu będzie mapa -->
</div>
• zainicjalizować mapę w funkcji JavaScript:
var wspolrzedne = new google.maps.LatLng(53.419,14.581);
var opcjeMapy = {
    zoom: 10,
    center: wspolrzedne,
    mapTypeId: google.maps.MapTypeId.ROADMAP
};
var mapa = new
    google.maps.Map(document.getElementById("mapka"),
        opcjeMapy);
```



Fundusze Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny

**LABORATORIUM 12. CZĘŚĆ 1. WYKORZYSTANIE ELEMENTÓW  
PROGRAMOWANIA OBIEKTOWEGO W JAVASCRIPT.  
REALIZACJA APLIKACJI PRACUJĄCEJ Z DANYMI  
W FORMACIE JSON.**

**Cel laboratorium:**

Celem zajęć jest poznanie i wykorzystanie dodatkowych możliwości programowania obiektowego w JavaScript.

**Zakres tematyczny zajęć:**

Zdefiniowanie klasy umożliwiającej zarządzanie rejestracją użytkownika.

Utworzenie strony z formularzem rejestracji użytkownika.

Sprawdzanie niepowtarzalności danych rejestracyjnych.

**Pytania kontrolne:**

1. Co to są funkcje anonimowe i kiedy mogą być wykorzystane? Czym są funkcje strzałkowe (wyrażenia lambda)?
2. Do czego służy metoda **addEventListener** i dla jakich obiektów może być stosowana?
3. Jak definiuje się własną klasę w JavaScript?

**Zadanie 12.1. Definicja klasy User**

1. Utwórz nowy dokument HTML o nazwie np. **user.html** i dodaj do niego skrypt z definicją klasy **User**, jak pokazano na listingu 12.1. Na listingu w bloku skryptu dodano również instrukcje do wykonania, ale dopiero po zajściu zdarzenia **DOMContentLoaded**, czyli pobraniu całego drzewa dokumentu HTML. Zrealizowane jest to za pomocą dodania funkcji słuchacza zdarzeń – **addEventListener**, którego pierwszym parametrem jest nazwa zdarzenia a drugim – funkcja do obsługi tego zdarzenia. Może to być **funkcja anonimowa** (jak w przykładzie). Zastosowano tutaj skrótny zapis z wyrażeniem lambda (więcej szczegółów znajdziesz w materiałach wykładowych).

**Listing 12.1. Plik user.html**

```
<!DOCTYPE html>
<html>
  <head>
    <title>Formularz rejestracji</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script>
      //definicja klasy User
      class User{
        constructor(email="", login="user", pass="user") {
          this.login=login;
          this.pass=pass;
          this.email=email;
        }
        pokaz(){
          //uzupełnij dane o loginie I emailu:
        }
      }
    </script>
  </head>
  <body>
    <h1>Witaj w naszej aplikacji!</h1>
    <form>
      <label>Email:</label>
      <input type="text" id="email" value="user@example.com" />
      <label>Login:</label>
      <input type="text" id="login" value="user" />
      <label>Hasło:</label>
      <input type="password" id="pass" value="user" />
      <button type="button" id="register">Zarejestruj się</button>
    </form>
  </body>
</html>
```



Fundusze Europejskie  
Wiedza Edukacja Rozwój



Rzeczypospolita  
Polska



```
        return "Dane użytkownika: login:"+ "...";
    }
} // koniec definicji klasy User

//skrypt główny - instrukcje są wykonywane dopiero po załadowaniu DOM:
document.addEventListener('DOMContentLoaded', () => {
    var user=new User();
    //pokaż dane o użytkowniku na konsoli

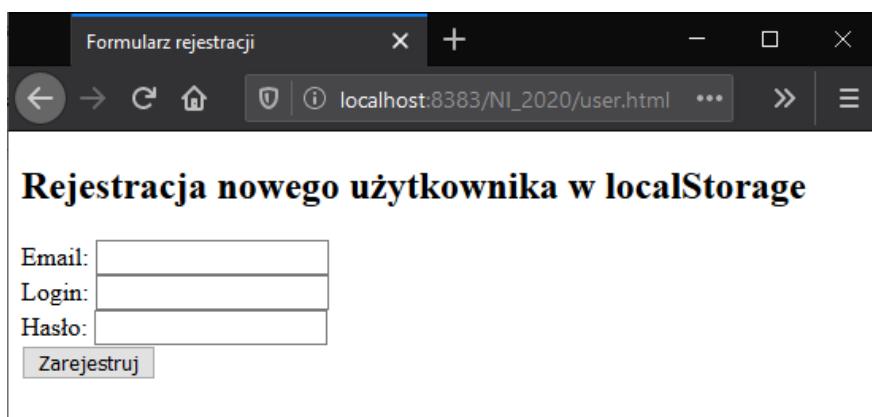
    });
</script>
</head>
<body>
    <div>
        <h2>Rejestracja nowego użytkownika w localStorage</h2>
        <!-- Przygotowanie elementów do wyświetlania formularza rejestracji i danych i komunikatów informacyjnych -->
        <div id='formularz'></div>
        <div id='info'></div>
    </div>
</body>
</html>
```

Uzupełnij fragmenty kodu z listingu 12.1, a następnie przetestuj działanie metody do wyświetlenia danych o użytkowniku w konsoli.

2. Do klasy dodaj metodę *formularzRejestracji()* (Listing 12.2), której zadaniem będzie wyświetlenie formularza do rejestracji nowego użytkownika (Rys. 12.1) w przygotowanym już bloku *<div id='formularz'>*. W skrypcie głównym wywołaj tę metodę, tak aby uzyskać efekt jak na rysunku 12.1.

**Listing 12.2. Fragment metody *formularzRejestracji()* z klasy User**

```
formularzRejestracji(){
    var formularz = "";
    formularz += '<div>email: uzupełnij </div>';
    return formularz;
}
```



Rys. 12.1. Widok strony z formularzem



Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska

Unia Europejska  
Europejski Fundusz Społeczny



3. Po kliknięciu na przycisk **Zarejestruj** (Listing 12.3), w skrypcie głównym należy:
  - a) pobrać dane z formularza i wykorzystać je do **zaktualizowania** pól obiektu **user** (utworzonego już wcześniej w skrypcie głównym);
  - b) **zapisać obiekt user w localStorage** pod kluczem o wartości loginu;
  - c) sprawdzić poprawność zapisu w **localStorage**.

**Listing 12.3. Obsługa kliknięcia na przycisk ze słuchaczem zdarzenia**

```
...// obsługa akcji kliknięcia na przycisk z id='rejestruj'  
rejestruj.addEventListener("click", ()=> {  
    user.login=document.getElementById('login').value;  
    //zbierz resztę danych z pól formularza  
    //zapisz obiekt user do localStorage...  
});
```

The screenshot shows a web browser window titled "Formularz rejestracji". The address bar indicates the page is at `localhost:8383/NI_2020/user.html`. The main content area displays a registration form with fields for Email, Login, and Hasło, and a "Zarejestruj" button. Below the form, a message states: "Istnieje już użytkownik o podanym loginie." (There is already a user with the specified login). The bottom part of the screenshot shows the developer tools' Application tab, specifically the Local Storage panel. It lists three items in the storage:

Key	Value
beata	{"login": "beata", "pass": "beata", "email": "beata@gmail.com"}
admin	{"login": "admin", "pass": "admin", "email": "admin@gmail.com"}
ania	{"login": "ania", "pass": "ania123", "email": "ania@onetpl"}

Below the table, a expanded view of the "ania" entry shows its details:

```
{login: "ania", pass: "ania123", email: "ania@onet.pl"}  
email: "ania@onet.pl"  
login: "ania"  
pass: "ania123"
```

Rys. 12.2. Widok strony z localStorage i komunikatem o istniejącym już użytkowniku

### **Zadanie 12.2. Sprawdzenie niepowtarzalności danych nowego użytkownika**

Uzupełnij skrypt główny dodatkowymi instrukcjami:

- a) dane użytkownika mogą być zapisane w **localStorage**, ale tylko po sprawdzeniu, czy nie istnieje tam już użytkownik o danym loginie. Jeśli taki **login** już istnieje, to należy wyświetlić odpowiedni komunikat (Rys.12.2).
- b) uzupełnić warunek sprawdzający – zapis nowego użytkownika może być zrealizowany, jeśli w bazie nie istnieje już użytkownik o takim samym loginie lub **emailu**.

### **Zadanie 12.3.**

Na podstawie przykładu klasy **Organizer** z wykładu – zmodyfikuj skrypt z zadania 11.3, tak aby zamiast funkcji operujących na liście produktów, wykorzystać odpowiednio zdefiniowaną klasę i jej metody.



**Fundusze Europejskie**  
Wiedza Edukacja Rozwój



**Rzeczpospolita Polska**



**Unia Europejska**  
Europejski Fundusz Społeczny

**LABORATORIUM 12. CZĘŚĆ 2. WYKORZYSTANIE ELEMENTÓW  
PROGRAMOWANIA OBIEKTOWEGO W JAVASCRIPT.  
REALIZACJA APLIKACJI PRACUJĄCEJ Z DANYMI  
W FORMACIE JSON.**

**Cel laboratorium:**

Celem zajęć jest zastosowanie programowania obiektowego w JavaScript do pracy w trybie asynchronicznym (przesyłania żądań HTTP w tle) z danymi w formacie JSON.

**Zakres tematyczny zajęć:**

Przygotowanie środowiska pracy – pakiet XAMPP i serwer lokalny Apache.

Przygotowanie projektu testowego na serwerze Apache.

Praca z danymi w formacie JSON w trybie asynchronicznym.

**Pytania kontrolne:**

1. Na czym polega praca w trybie asynchronicznym?
2. Co to jest AJAX i obiekt XMLHttpRequest (XHR)?
3. Co to jest Fetch API?
4. Wymień i scharakteryzuj podstawowe interfejsy Fetch API.
5. Czym są obiekty Promise i jak się z nich korzysta w asynchronicznym przesyłaniu danych?

**Wprowadzenie**

Aby można było przetestować pracę w trybie asynchronicznym za pomocą AJAX, potrzebujemy serwera, obsługującego żądania HTTP. Do tego celu wykorzystamy serwer **Apache** dostępny po zainstalowaniu pakietu XAMPP na wybranym systemie operacyjnym (Windows/Linux/Mac – do pobrania ze strony <https://www.apachefriends.org/pl/download.html>). Po instalacji pakietu:

1. Wyszukaj folder pakietu (**xampp**) oraz przejrzyj zawartość podfolderu **htdocs**. Jeśli korzystasz z innego pakietu lub samodzielnie konfigurujesz środowisko pracy – postępowanie jest analogiczne).
2. Uruchom serwer Apache (w przypadku XAMPP możesz skorzystać z **XAMPP Control Panel – run Server**) (Rys. 12.3).
3. Sprawdź, czy serwer działa prawidłowo - w przeglądarce wpisz adres, np. **http://localhost/** (ewentualnie będzie dodatkowo potrzebny numer portu, na którym działa serwer Apache np. **http://localhost:80/**). Domyślnie zostanie uruchomiony skrypt *index.php* z folderu **htdocs** (Rys. 12.4).



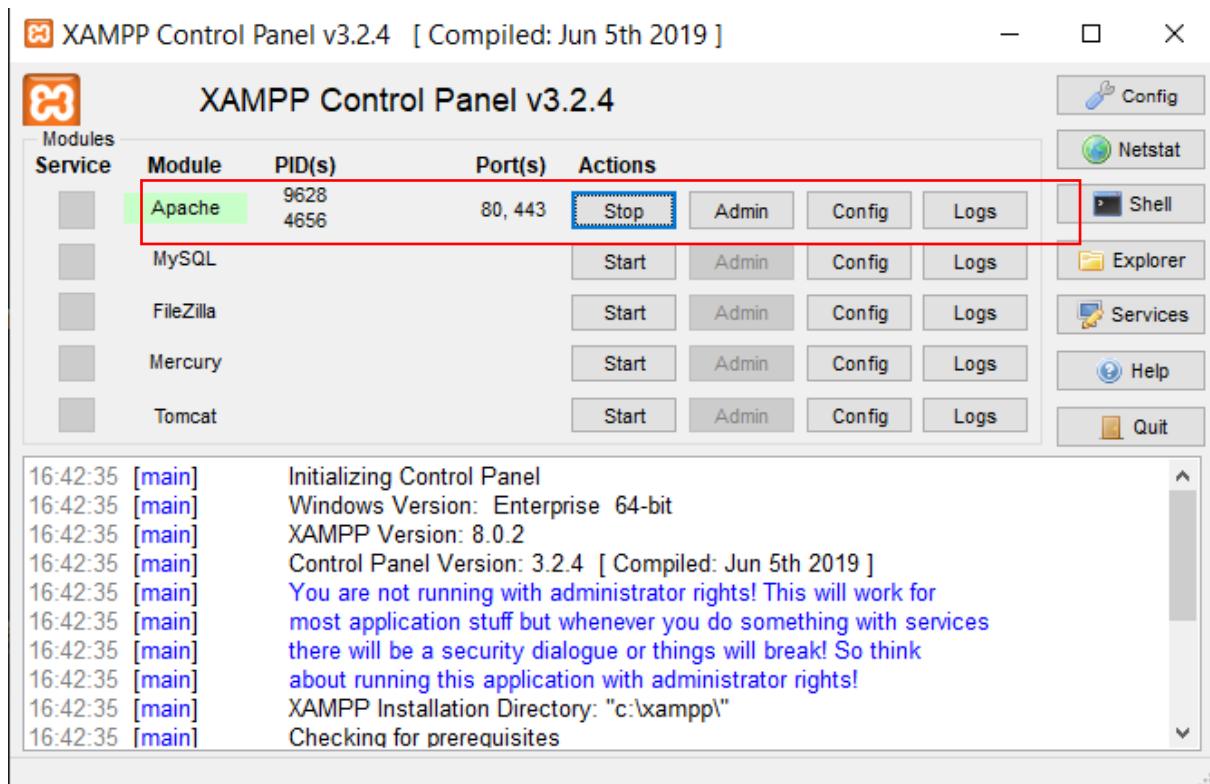
Fundusze Europejskie  
Wiedza Edukacja Rozwój



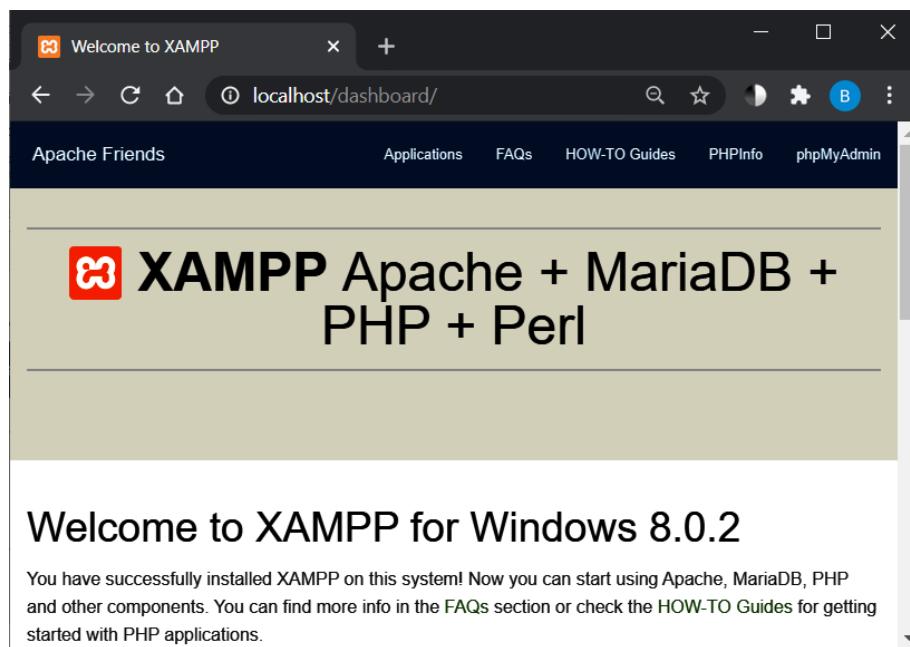
Rzeczpospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny



Rys. 12.3. Okienko Control Panel dla XAMPP



Rys. 12.4. Widok po uruchomieniu skryptu index.php na hoście lokalnym

#### Zadanie 12.4. Pobieranie danych z serwera w trybie asynchronicznym za pomocą obiektu XMLHttpRequest

- Na serwerze w katalogu `c:\xampp\htdocs` utwórz folder `testajax`. Umieść w nim dokument `index.html` z listingu 12.4 oraz folder `css` z plikiem `style.css` z listingu 12.5.

**Listing 12.4. Plik index.html**

```
<!DOCTYPE html>
<html>
    <head>
        <title>Test Ajax</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <link rel="stylesheet" href="css/style.css" />
    </head>
    <body>
        <header>
            <h1>Test AJAX</h1>
            <nav>
                <button id="b1" onclick="pobierzDane('info')">0 stronie</button>
                <button id="b2" onclick="pobierzDane('act')">Aktualności</button>
                <button id="b3">Galeria</button>
                <button id="b4">Formularz</button>
            </nav>
        </header>
        <section id="s1">
            <h2>0 stronie</h2>
            <p>
                <b>AJAX </b>(Asynchronous JavaScript and XML) ...
            </p>
        </section>
        <footer>&copy;BP</footer>
        <script src="js/skrypty.js"></script>
    </body>
</html>
```

**Listing 12.5. Plik style.css**

```
body {
    background:#d4c7b9;
}
h1,h2,footer {
    padding:0.2em;
    text-align:center;
}
nav {
    float:left;
    width:100%;
}
nav button {
    background: #45525a;
    color:#ffe9ba;
    float:left;
    display:block;
    width:24%;
    margin: 0.5%;
    padding:1% 0;
    font-size:110%;
    text-align:center;
    border-radius:20px;
}
footer {
    background: #45525a;
```



**Fundusze Europejskie**  
Wiedza Edukacja Rozwój

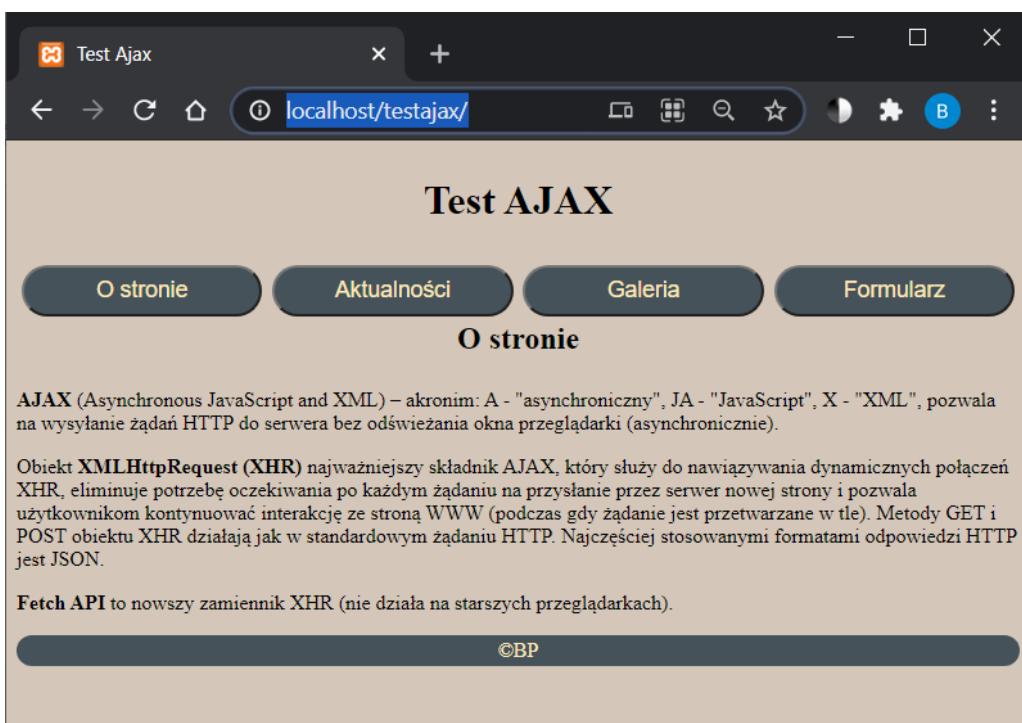


**Rzeczypospolita  
Polska**



```
color:#ffe9ba;
clear:both;
border-radius:20px;
}
@media screen and (max-width: 720px) {
nav button {
width: 48%;
box-sizing: border-box;
padding: 10px 0;
font-size:120%;}
}
@media screen and (max-width: 560px) {
nav button {
width: 98%;
box-sizing: border-box;
padding: 2% 0;
font-size:140%;}
}
```

2. Plik **index.html** jest domyślnie stroną startową projektu. Uruchom stronę z hosta lokalnego (*localhost*), wpisując w przeglądarce adres: <http://localhost/testajax/> (Rys. 12.5). Pamiętaj, że **serwer Apache musi być uruchomiony**.



Rys. 12.5. Plik index.html w przeglądarce

3. Korzystając z kodu na listingu 12.6 i materiałów wykładowych, uzupełnij kod skryptu JS tak, aby za pomocą AJAX następowała wymiana zawartości bloku sekcji o *id='s1'* (na tekst pobrany z odpowiedniego pliku z folderu **dane** – Rys. 12.6), w odpowiedzi na kliknięcie przycisku w elemencie nawigacyjnym. Na serwerze utwórz katalog **dane**, a w nim cztery pliki tekstowe z odpowiednią treścią (treść może zawierać tagi HTML).



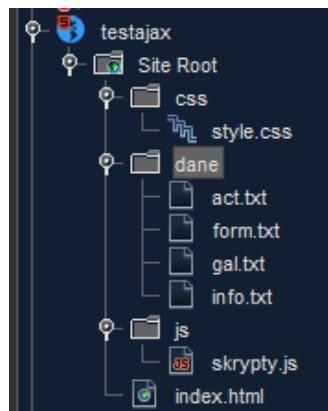
Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny



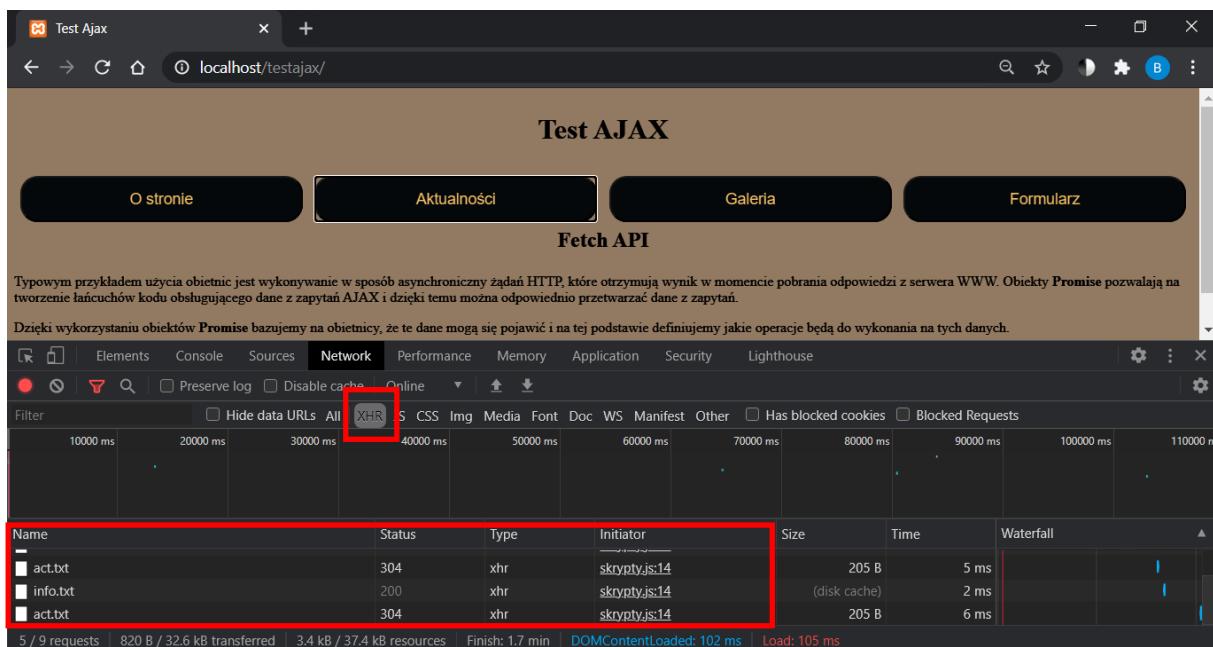
Rys. 12.6. Zasoby projektu

### **Listing 12.6. Skrypt skrypty.js**

```
//utworzenie obiektu xhr:
const xhr = new XMLHttpRequest();

function pobierzDane(nazwaPliku)
{   if (xhr) {
        var url = "http://localhost/testajax/dane/" + nazwaPliku + ".txt";
        xhr.open("GET", url);
        xhr.addEventListener("readystatechange", function () {
            if (xhr.readyState === 4) {
                document.getElementById("s1").innerHTML = xhr.responseText;
            }
        });
        xhr.send(null);
    }
}
```

4. Zaobserwuj, korzystając z zakładki **Network** w przeglądarce Chrome, jak są przesyłane żądania w trybie asynchronicznym z obiektem XHR (Rys. 12.7).



Rys. 12.7. Żądania przesyłane w trybie asynchronicznym XHR



**Fundusze Europejskie**  
Wiedza Edukacja Rozwój



**Rzeczpospolita  
Polska**

**Unia Europejska**  
Europejski Fundusz Społeczny



### Zadanie 12.5. Pobieranie danych z serwera za pomocą Fetch API

Zrealizuj to samo zadanie, ale korzystając z nowszego interfejsu **Fetch API** (Listing 12.7).

#### Listing 12.7. Schemat działania z Fetch API

```
//Skrypt z Fetch API
//sprawdź czy DOM został załadowany:
document.addEventListener("DOMContentLoaded", function() {
    //obsługa zdarzenia kliknięcia na b1:
    var but1 = document.getElementById("b1");
    but1.addEventListener('click', function(){
        fetch("http://localhost/testajax/dane/info.txt")
            .then( response => {return response.text();} )
            .then( dane => { document.getElementById("s1").innerHTML = dane; } )
    },
    false);

    //obsługa zdarzenia kliknięcia na b2:
    var but2 = document.getElementById("b2");
    but2.addEventListener('click', function(){
        //uzupełnij
    },
    false);
    //obsługa pozostałych zdarzeń:
    //uzupełnij
})
```

Zaobserwuj, korzystając z zakładki **Network** w przeglądarce Chrome, jak teraz są przesyłane żądania w trybie asynchronicznym (Rys. 12.8).

The screenshot shows the Network tab in the Chrome DevTools. The 'Type' column is highlighted with a red box, showing two entries: 'fetch' for each request. The requests are identified as 'act.txt' and 'info.txt'.

Name	Status	Type	Initiator	Size	Time	Waterfall
act.txt	200	fetch	skrypty.js:32	(disk cache)	3 ms	
info.txt	200	fetch	skrypty.js:24	(disk cache)	3 ms	

Rys. 12.8. Żądania przesyłane w trybie asynchronicznym z Fetch API



Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



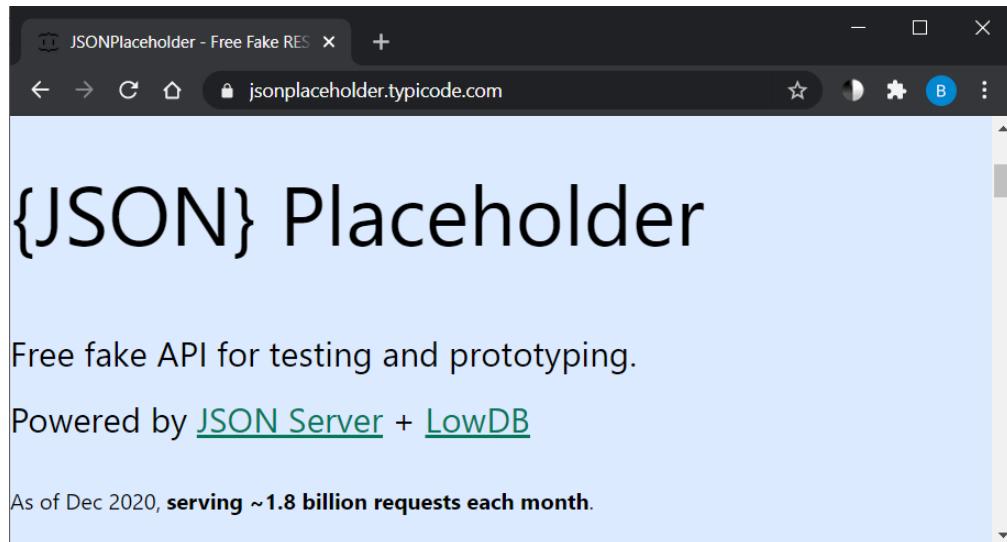
Rzeczpospolita  
Polska



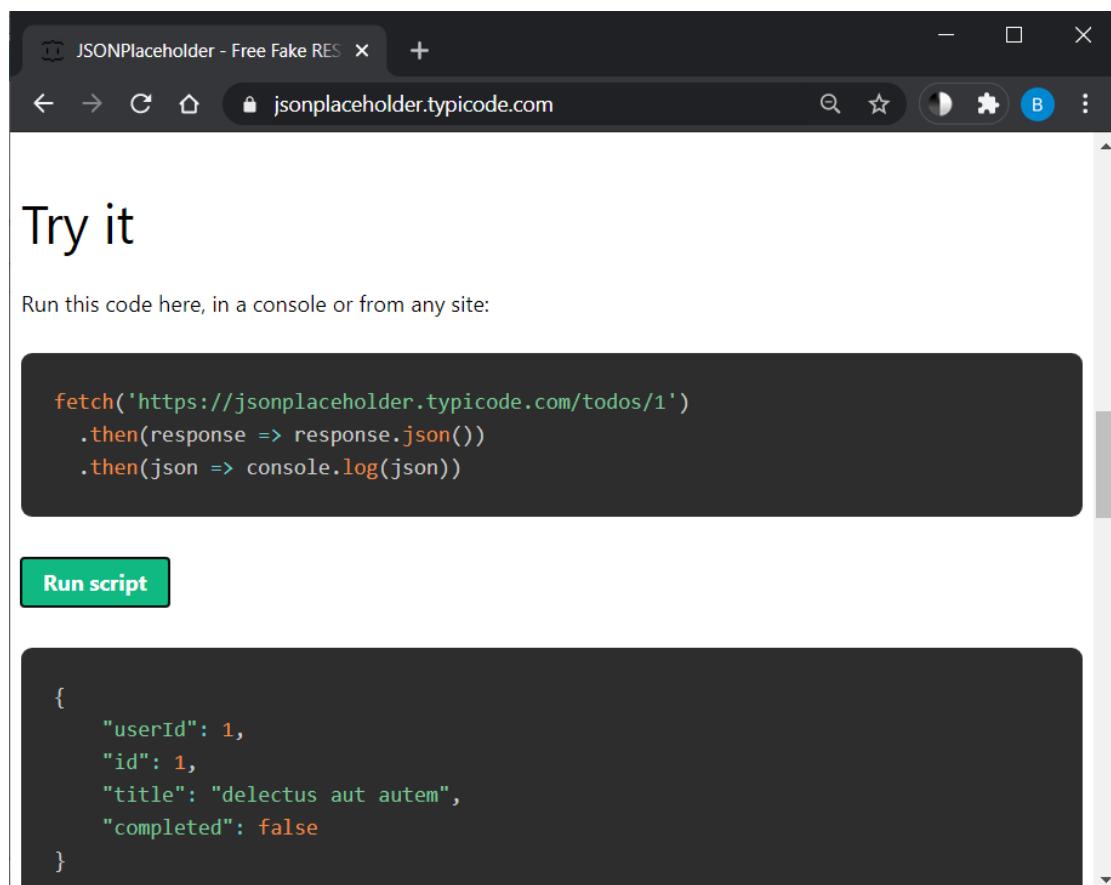
Unia Europejska  
Europejski Fundusz Społeczny

### Zadanie 12.6. Praca z JSON. Wysyłanie danych na serwer

Ponieważ nie mamy jeszcze własnego skryptu na serwerze, żeby przetestować wysyłanie danych - skorzystaj ze strony: <https://jsonplaceholder.typicode.com/> (Rys. 12.9) i za jej pomocą sprawdź jak pracować z danymi w formacie JSON i wykorzystać **Fetch API** do przesyłania danych na serwer (Rys. 12.10, 12.11).



Rys. 12.9. Strona do testów przesyłania danych JSON



Rys. 12.10. Metoda GET i Ferch API



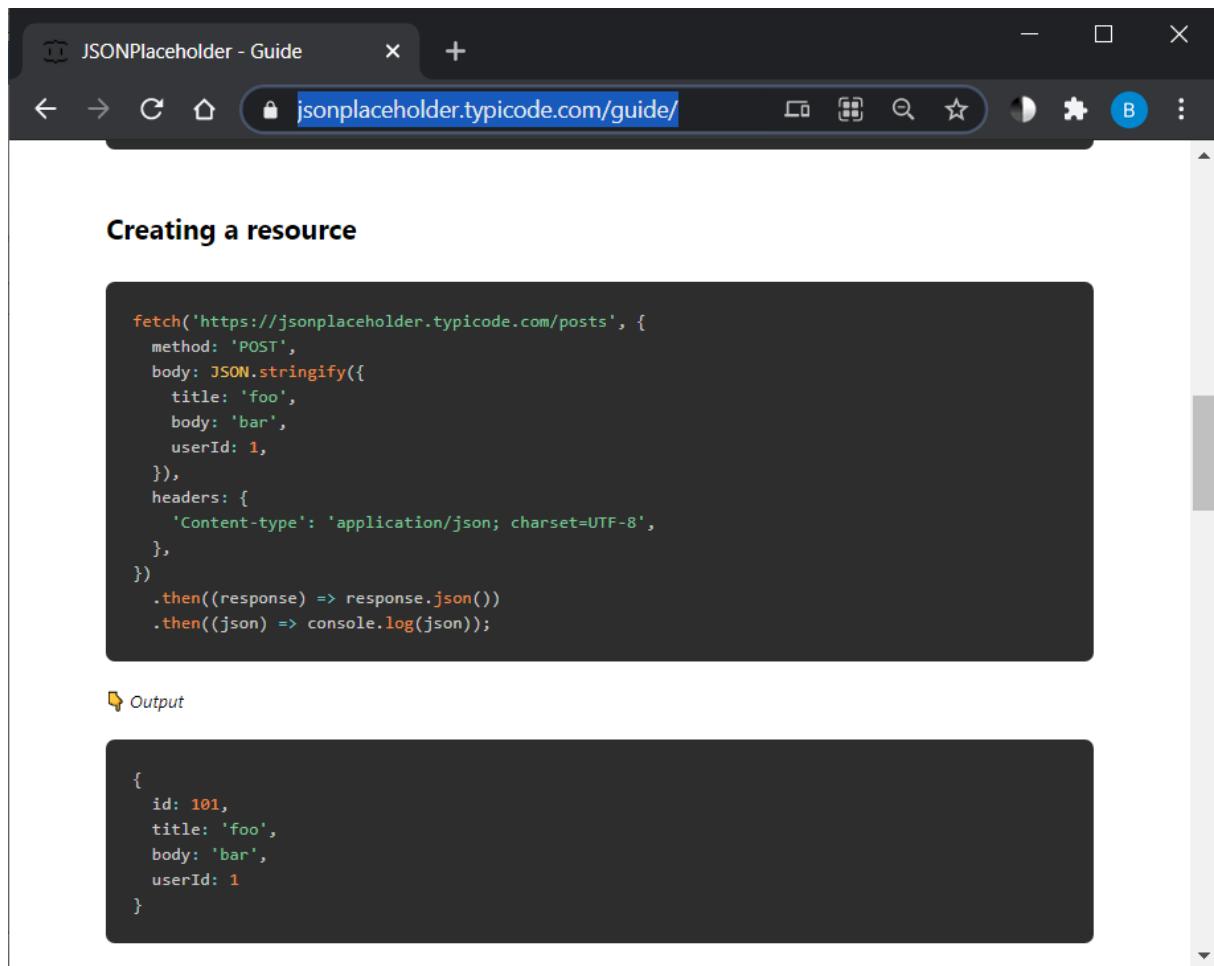
Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska

Unia Europejska  
Europejski Fundusz Społeczny





The screenshot shows a browser window with the title "JSONPlaceholder - Guide". The address bar contains "jsonplaceholder.typicode.com/guide/". The main content area displays a heading "Creating a resource" followed by a code snippet in a dark-themed code editor. The code uses the Fetch API to send a POST request to "https://jsonplaceholder.typicode.com/posts". It includes a JSON payload with fields title, body, and userId, and sets the Content-type header to "application/json; charset=UTF-8". The response is captured using .then() and logged to the console. Below the code editor, there is an "Output" section showing the JSON response object:

```
fetch('https://jsonplaceholder.typicode.com/posts', {
  method: 'POST',
  body: JSON.stringify({
    title: 'foo',
    body: 'bar',
    userId: 1,
  }),
  headers: {
    'Content-type': 'application/json; charset=UTF-8',
  },
})
.then((response) => response.json())
.then((json) => console.log(json));
```

Output

```
{  
  id: 101,  
  title: 'foo',  
  body: 'bar',  
  userId: 1  
}
```

Rys. 12.11. Metoda POST i Fetch API



Fundusze Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska



Unia Europejska  
Europejski Fundusz Społeczny

## **LABORATORIUM 13. PREZENTACJA PROJEKTÓW ZALICZENIOWYCH**

### **Cel laboratorium:**

Celem zajęć jest prezentacja przez studentów projektów aplikacji internetowych wykonanych zgodnie z zadaną specyfikacją na zaliczenie przedmiotu.

### **Zakres tematyczny zajęć:**

Prezentacja projektów przez studentów.

Dyskusja na temat wykorzystanych rozwiązań.

Ocena projektów.



Materiały zostały opracowane w ramach projektu  
„Zintegrowany Program Rozwoju Politechniki Lubelskiej – część druga”,  
umowa nr **POWR.03.05.00-00-Z060/18-00**  
w ramach Programu Operacyjnego Wiedza Edukacja Rozwój 2014-2020  
współfinansowanego ze środków Europejskiego Funduszu Społecznego



Fundusze  
Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska

Unia Europejska  
Europejski Fundusz Społeczny

