

Aplikacje internetowe - laboratorium

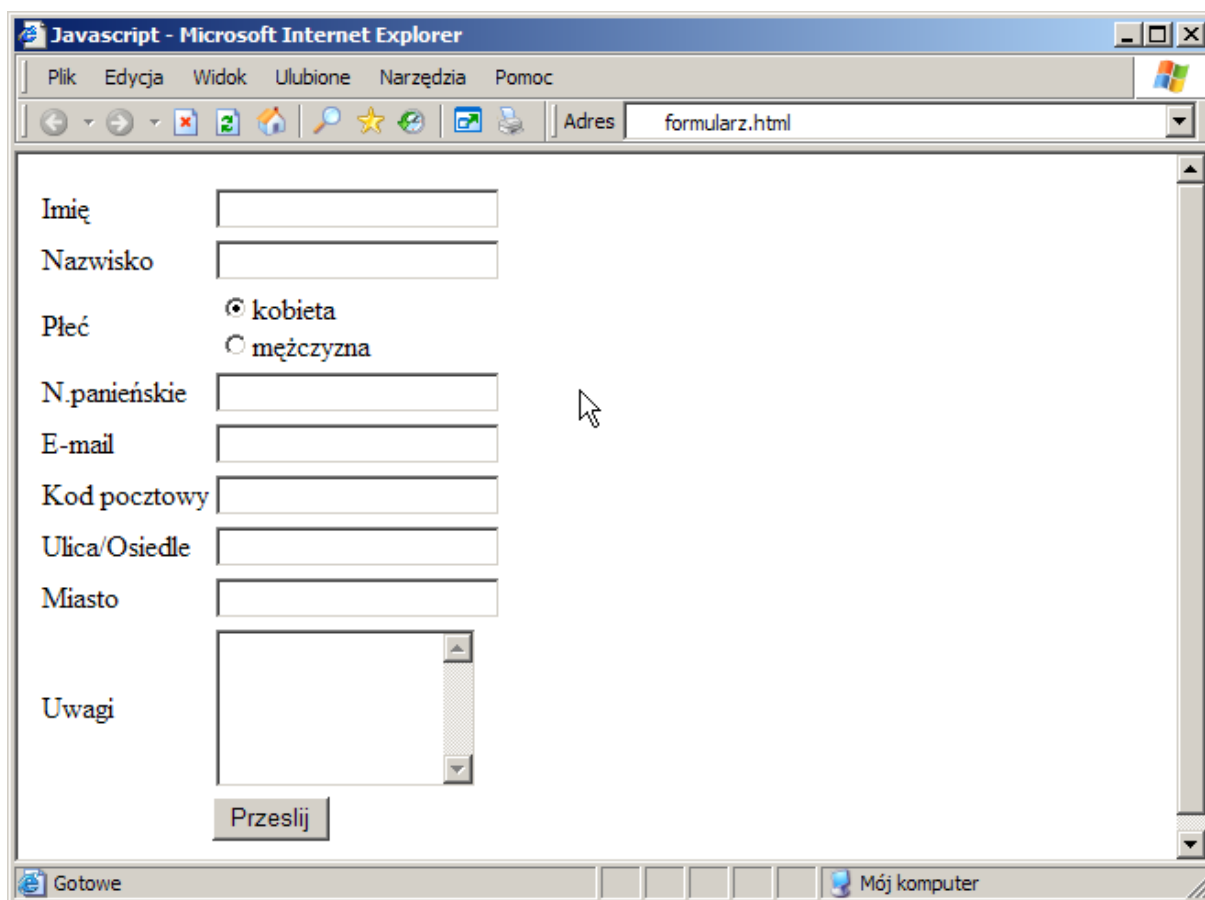
Język JavaScript

Celem ćwiczenia jest przygotowanie formularza HTML z wykorzystaniem języka JavaScript. Formularz ten będzie sprawdzany pod względem zawartości przed wysłaniem do serwera. Formularz będzie miał charakter dynamiczny, tzn. niektóre jego elementy będą zmieniać swój stan pod wpływem działań użytkownika. Do wykonania ćwiczenia potrzebny jest dowolny edytor plików tekstowych oraz przeglądarka.

1. Stwórz dwa pliki tekstowe znajdujące się w tym samym katalogu dyskowym: `formularz.html` i `form_check.js`.
2. Formularz ma służyć do wprowadzania danych potrzebnych do rejestracji użytkownika w serwisie internetowym. Potrzebne informacje to przede wszystkim dane osobowe. W celu utworzenia formularza wykorzystaj element FORM języka HTML. Formularz o nazwie `dane_osobowe` powinien umożliwiać wprowadzanie następujących danych: imię, nazwisko, płeć (wybór jednej z opcji), nazwisko panińskie, e-mail, ulica, kod pocztowy, miasto, uwagi (pole tekstowe). Dla lepszej wizualizacji formularza pola można umieścić w komórkach tabeli. Zawartość pliku `formularz.html` powinna być następująca:

```
<html>
<head>
  <title>Javascript</title>
  <meta http-equiv="Content-Type" content="text/html; charset=windows-1250">
</head>
<body>
  <form name="dane_osobowe">
    <table border=0>
      <tr><td>Imię</td><td><input type="text" name="f_imie"></td></tr>
      <tr><td>Nazwisko</td><td><input type="text" name="f_nazwisko"></td></tr>
      <tr><td>Płeć</td><td>
        <input name="f_plec" value="f_k" checked type="radio"/>kobieta<BR>
        <input name="f_plec" value="f_m" type="radio"/>mężczyzna</td></tr>
      <tr><td>N.panińskie</td><td><input type="text" name="f_nazwisko_p"></td></tr>
      <tr><td>E-mail</td><td><input type="text" name="f_email"></td></tr>
      <tr><td>Kod pocztowy</td><td><input type="text" name="f_kod"></td></tr>
      <tr><td>Ulica/Osiedle</td><td><input type="text" name="f_ulica"></td></tr>
      <tr><td>Miasto</td><td><input type="text" name="f_miasto"></td></tr>
      <tr><td>Uwagi</td><td>
        <textarea rows="5" cols="15" name="uwagi"></textarea></td></tr>
      <tr><td colspan="2" align="center">
        <input type="button" value="Przeslij"></td></tr>
    </table>
  </form>
</body>
</html>
```

Plik jest wyświetlany przez przeglądarkę w następujący sposób



3. W pliku `form_check.js` zostaną umieszczone definicje funkcji, które będą sprawdzały elementarne warunki, jakie powinny spełniać wartości wprowadzane do formularza. Sprawdzana powinna być długość napisu, format kodu pocztowego oraz adresu poczty internetowej. Początkowo, plik zawiera jedną funkcję sprawdzającą, czy pole jest puste.

```
// zwraca wartosc true jesli przekazany argument to pusty lancuch  
function isEmpty(str)  
{  
  if (str.length == 0)  
  {  
    return true  
  }  
  else  
  {  
    return false  
  }  
}
```

4. Dodaj do powyższego kodu funkcję, która sprawdzi, czy zgłaszając formularz użytkownik wypełnił wymagane pole „Imię”

```
function validate(form)
{
    if (isEmpty(form.elements["f_imie"].value))
    {
        alert("Podaj imię!")
        return false
    }
    else
    {
        return true
    }
}
```

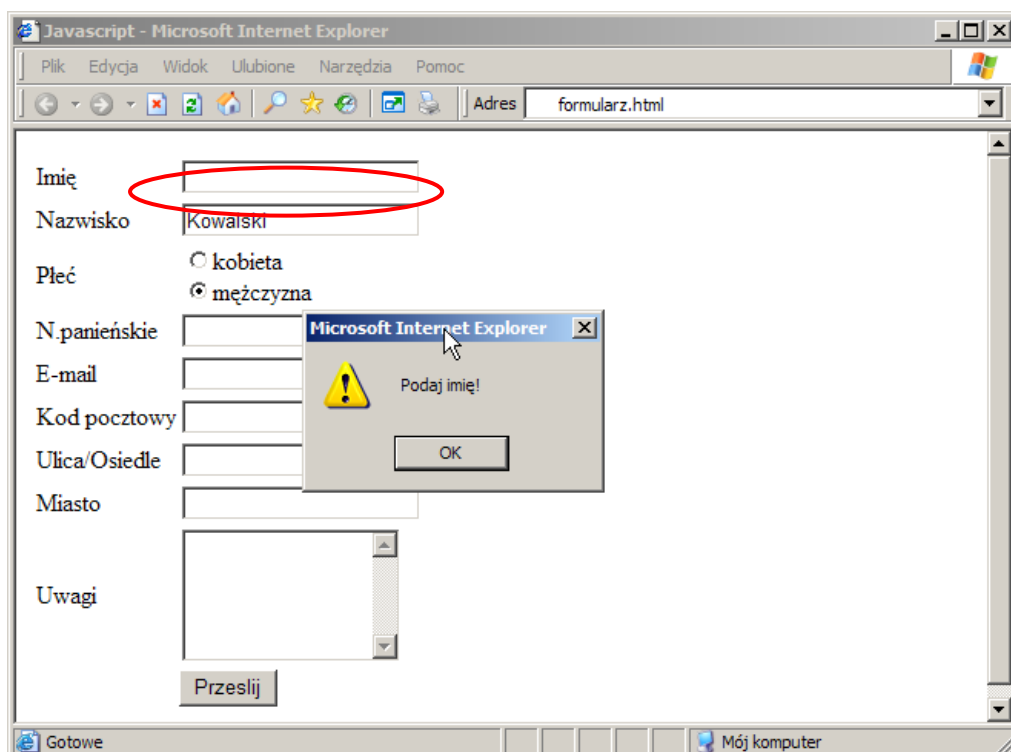
5. Dodaj do nagłówka HTML w pliku formularz.html odwołanie do zewnętrznego pliku (form_check.js) zawierającego skrypt w języku JavaScript:

```
<html>
<head>
    <title>Javascript</title>
    <meta http-equiv="Content-Type" content="text/html; charset=windows-1250">
    <script src="form_check.js"></script>
</head>
<body>
    ...
```

6. Dodaj do przycisku obsługę zdarzenia naciśnięcia przycisku. Właściwy fragment kodu powinien mieć postać:

```
<input type="button" value="Przeslij"
      onClick="return validate(this.form)">
```

7. Uruchom formularz w przeglądarce. Przetestuj działanie formularza próbując zgłosić formularz bez wypełnienia pola „Imię”.



8. Formularz nadal można łatwo oszukać, wpisując do pola „Imię” ciąg białych znaków. Dodaj do pliku `form_check.js` funkcję sprawdzającą, czy podany łańcuch znaków nie składa się w całości z białych znaków.

```
// zwraca wartosc true jesli przekazany argument
// to ciag bialych znakow
function isWhiteSpace(str)
{
    var ws = "\t\n\r "
    for (i = 0; i < str.length; i++)
    {
        var c = str.charAt(i)
        if ( ws.indexOf(c) == -1)
            return false
    }
    return true
}
```

9. Zmodyfikuj funkcję `validate()` w taki sposób, aby przy sprawdzaniu imienia uwzględnić także funkcję `isWhiteSpace()`.

```
function validate(form)
{
    if (isEmpty(form.elements["f_imie"].value) ||
        isWhiteSpace(form.elements["f_imie"].value))
    {
        alert("Podaj imię!")
        return false
    }
    else
        return true
}
```

10. Sprawdź, czy walidacja działa poprawnie.

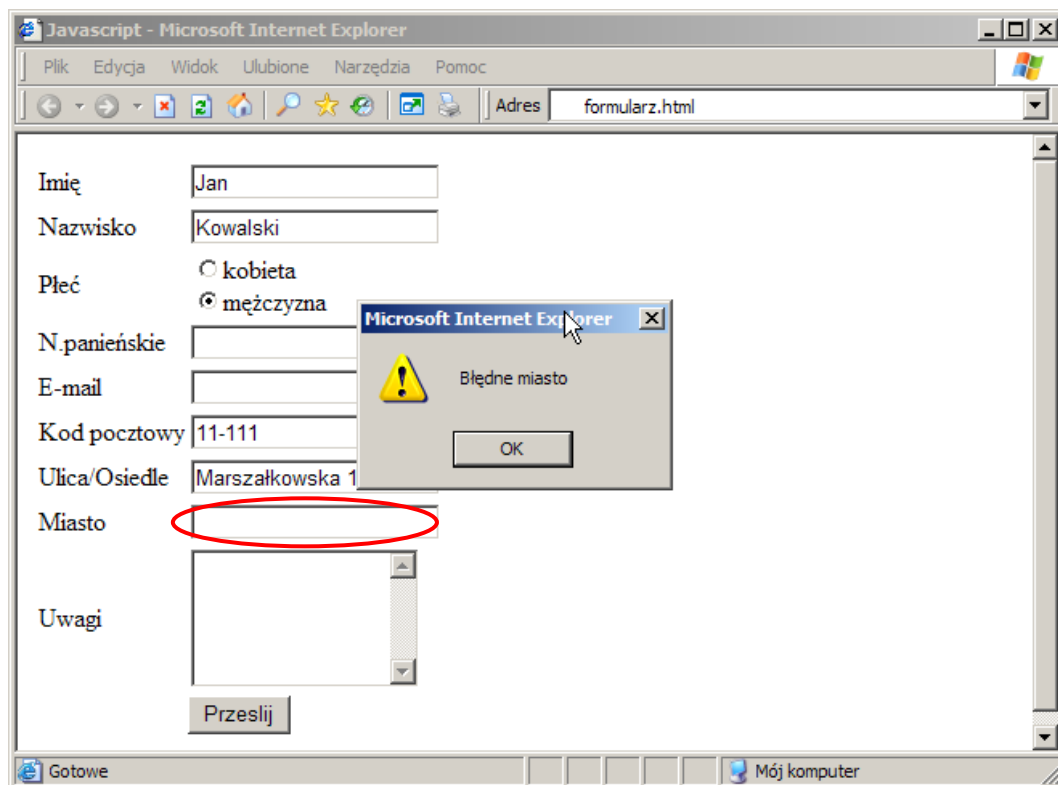
11. Zmodyfikujemy teraz kod skryptu w taki sposób, aby wygodnie można było dodawać walidację poprawności kolejnych pól. Dodaj do pliku `form_check.js` funkcję `checkString()`

```
// zwraca wartosc prawda jesli przekazany argument to niepusty lancuch,
// ktory nie zawiera samych bialych znakow
function checkString(str, msg)
{
    if ( isWhiteSpace(str) || isEmpty(str))
    {
        alert(msg)
        return false
    }
    else
        return true
}
```

12. Zmień funkcję `validate()` dodając sprawdzanie poprawności nazwiska, ulicy i miasta.

```
function validate(form)
{
    return (
        checkString(form.elements["f_imie"].value, 'Błędne imię') &&
        checkString(form.elements["f_nazwisko"].value, ' Błędne nazwisko') &&
        checkString(form.elements["f_kod"].value, ' Błędny kod pocztowy') &&
        checkString(form.elements["f_ulica"].value, ' Błędna ulica/osiedle') &&
        checkString(form.elements["f_miasto"].value, ' Błędne miasto') )
}
```

13. Sprawdź działanie formularza, spróbuj przesłać formularz bez podania nazwy ulicy lub nazwy miasta



14. W następnym kroku uzupełnimy skrypt o funkcję weryfikującą podstawową poprawność podanego adresu e-mail. Dodaj do pliku `form_check.js` poniższy kod.

```
// zwraca wartosc prawda jesli przekazany
// argument to poprawny adres email
function checkEmail(str)
{
    if (isWhiteSpace(str))
        alert("Podaj właściwy e-mail")
    else
    {
        at = str.indexOf("@")
        if (at < 1)
        {
            alert("Nieprawidłowy e-mail")
            return false
        }
        else
        {
            var l = -1
            for (var i = 0; i < str.length; i++)
            {
                var c = str.charAt(i)
                if ( c == ".")
                    l = i
            }
            if ( (l < (at+2) ) || (l == str.length-1) )
                alert("Nieprawidłowy e-mail")
        }
        return true
    }
}
```

15. Aby uruchomić weryfikację adresu email, konieczna jest modyfikacja funkcji `validate()`

```
function validate(form)
{
    return (
        checkString(form.elements["f_imie"].value, 'Błędne imię') &&
        checkString(form.elements["f_nazwisko"].value, 'Błędne nazwisko') &&
        checkEmail(form.elements["f_email"].value) &&
        checkString(form.elements["f_kod"].value, 'Błędny kod pocztowy') &&
        checkString(form.elements["f_ulica"].value, 'Błędna ulica/osiedle') &&
        checkString(form.elements["f_miasto"].value, 'Błędne miasto') )
}
```

16. Przedstawione rozwiązanie nie jest zadowalające, ponieważ zmusza użytkownika do dwukrotnego kliknięcia myszką (zamknięcie okna z powiadomieniem oraz przejście do błędnie wypełnionego pola). Następny przykład pokaże, jak temu zaradzić przy użyciu liczników czasowych. Zmodyfikuj formularz poprzez dodanie po polu do wprowadzania imienia pustego pola do wyświetlania błędów. Zwróć uwagę, aby nazwa pola do wyświetlania błędów różniła się od nazwy pola do wprowadzania danych tylko pierwszą literą.

```
...  
<table border=0>  
  <tr>  
    <td>Imię</td>  
    <td><input type=text name="f_imie"><span class="err" id="e_imie"/></td>  
  </tr>
```

17. Dodaj w nagłówku regułę CSS do poprawnego formatowania komunikatu o błędzie

```
<style>  
  span.err { color : red; font-weight : bold; padding-left : 5px }  
</style>
```

18. Dodaj definicję nowej funkcji do weryfikacji pola

```
function checkStringAndFocus(obj, msg)  
{  
  var str = obj.value  
  var errorFieldName = "e_" + obj.name.substr(2,obj.name.length)  
  if ( isWhiteSpace(str) || isEmpty(str))  
  {  
    document.getElementById(errorFieldName).innerHTML = msg  
    obj.focus()  
    return false  
  }  
  else  
    return true  
}
```

19. Konieczna jest także modyfikacja funkcji validate()

```
function validate(form)  
{  
  return (  
    checkStringAndFocus(form.elements["f_imie"], 'Błędne imię') &&  
    checkString(form.elements["f_nazwisko"].value, 'Błędne nazwisko') &&  
    checkEmail(form.elements["f_email"].value) &&  
    checkString(form.elements["f_kod"].value, 'Błędny kod pocztowy') &&  
    checkString(form.elements["f_ulica"].value, 'Błędna ulica/osiedle') &&  
    checkString(form.elements["f_miasto"].value, 'Błędne miasto') )  
}
```

20. Niestety, tak zdefiniowany komunikat o błędzie jest wyświetlany nawet wówczas, gdy użytkownik poprawi już swój błąd, co może być mylące. Posłużymy się licznikiem czasowym do automatycznego usunięcia komunikatu o błędzie po upływie 5 sekund. Dodaj do pliku dwie funkcje: pierwsza funkcja powoduje uruchomienie stopera ustawionego na 5000 ms i automatyczne wywołanie drugiej funkcji po upływie określonego czasu. Nazwa pola, które ma zostać wyczyszczone, zostanie przekazane poprzez zmienną globalną.

```
errorField = ""

function startTimer(fName)
{
    errorField = fName
    window.setTimeout("clearError(errorField)", 5000)
}

function clearError(objName)
{
    document.getElementById(objName).innerHTML = ""
}
```

21. Ostatnim krokiem tej części ćwiczenia jest wskazanie miejsca, w którym funkcja `startTimer()` ma zostać uruchomiona. Zmodyfikuj definicję funkcji `checkStringAndFocus()` w następujący sposób.

```
...
var errorFieldName = "e_" + obj.name.substr(2,obj.name.length)
if ( isWhiteSpace(str) || isEmpty(str))
{
    document.getElementById(errorFieldName).innerHTML = msg
    startTimer(errorFieldName)
    obj.focus()
    return false
}
...
```

22. Uruchom formularz i przetestuj działanie mechanizmu raportowania błędów.

23. Kolejnym krokiem ćwiczenia będzie rozbudowanie formularza w taki sposób, aby pole „Nazwisko panięskie” było dostępne tylko dla osoby, która jest kobietą. W tym celu konieczne będzie obsłużenie zdarzenia polegającego na przełączeniu pomiędzy wartościami pola radiowego *kobieta/mężczyzna*. W zależności od kierunku zmiany pole „Nazwisko panięskie” powinno stać się widoczne lub niewidoczne. Do tego celu zostanie wykorzystana właściwość stylu o nazwie `visibility`. Może ona przyjąć wartości `visible` albo `hidden`. Zmiana nastąpi przy zajściu zdarzenia `onClick`.

24. Dodaj do pliku formularz.html tuż przed zamknięciem elementu body kod JavaScript odpowiedzialny za zmianę właściwości `visibility` dla elementu, którego identyfikator jest przekazywany do funkcji jako argument.

```
...  
<script type=text/javascript>  
    function showElement(e)  
    {  
        document.getElementById(e).style.visibility = 'visible';  
    }  
    function hideElement(e)  
    {  
        document.getElementById(e).style.visibility = 'hidden';  
    }  
</script>  
</body>  
...
```

25. Rozbuduj definicję pola służącego do wprowadzania nazwiska panińskiego w taki sposób, aby możliwe było jego ukrywanie. Dotychczasowy kod:

```
...  
<input type=text name="f_nazwisko_p">  
...
```

zastąp przez:

```
...  
<span id="NazwiskoPanienskie" style="visibility:visible;">  
    <input type=text name="f_nazwisko_p">  
</span>  
...
```

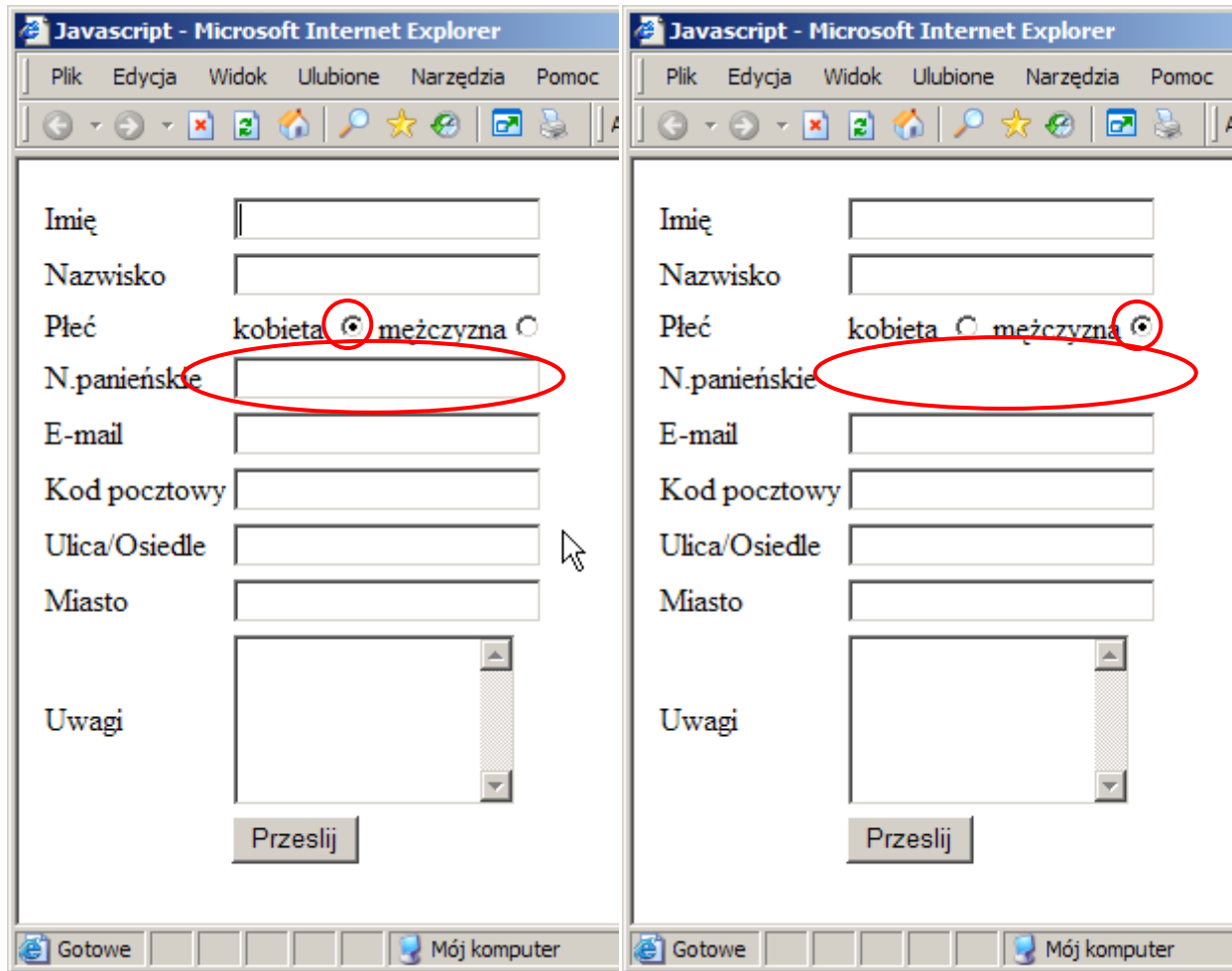
26. Dodaj obsługę zdarzenia polegającego na wybraniu jednej z opcji oznaczających płeć. W zależności od wybranej płci powinno nastąpić ukrycie albo wyświetlenie pola do wprowadzenia nazwiska panińskiego. Dotychczasowy kod:

```
...  
<input name="f_plec" value="f_k" checked type="radio"/>  
...  
<input name="f_plec" value="f_m" type="radio"/>  
...
```

zastąp przez:

```
...  
<input name="f_plec" value="f_k" checked type="radio"  
    onClick="showElement('NazwiskoPanienskie')"/>  
...  
<input name="f_plec" value="f_m" type="radio"  
    onClick="hideElement('NazwiskoPanienskie')"/>  
...
```

27. Uruchom formularz w przeglądarce i sprawdź jego działanie.



28. Język JavaScript oferuje możliwość wykorzystania mechanizmu wyrażeń regularnych. Mogą one służyć do weryfikacji czy dany napis odpowiada zdefiniowanemu wzorcowi. W utworzonym formularzu można znaleźć dwa fragmenty, gdzie warto skorzystać z wyrażeń regularnych. Jest to sprawdzanie poprawności kodu pocztowego oraz adresu mailowego. Dodaj do pliku `form_check.js` funkcję `checkEmailRegex()`. Następnie, zamień w funkcji `validate()` wywołanie funkcji `checkEmail()` na wywołanie funkcji `checkEmailRegex()`. Przetestuj działanie formularza.

```
// zwraca wartosc prawda jesli przekazany argument
// to poprawny adres email
function checkEmailRegex(str)
{
    var email = /^[a-zA-Z_0-9\.] +@[a-zA-Z_0-9\.] +\.[a-zA-Z][a-zA-Z]+/
    if (email.test(str))
        return true;
    else
        alert("Podaj właściwy e-mail");
}
```

29. Kolejnym zadaniem w ćwiczeniu będzie usprawnienie działania formularza przez dodanie weryfikacji kodu pocztowego w „czasie rzeczywistym”, tzn. podczas wpisywania tekstu przez użytkownika. W tym celu konieczne jest wykorzystanie zdarzenia następującego w chwili wpisania nowego znaku w pole kodu pocztowego.

Dotychczasowy kod

```
...  
<td><input type=text name="f_kod"></td>  
...
```

zastąp przez:

```
...  
<td width="200">  
  <input type=text name="f_kod" onKeyUp="checkZIPCodeRegEx(this);">  
  <b id="kod" class="none"></b>  
</td>  
...
```

Powyższa zmiana wprowadza obsługę zdarzenia `onKeyUp`, które następuje po zwolnieniu klawisza. Dodatkowo, obok pola do wprowadzania kodu pocztowego, został dodany tekst z identyfikatorem `kod`. Ten tekst będzie sygnalizował użytkownikowi, czy wpisywany kod jest poprawny.

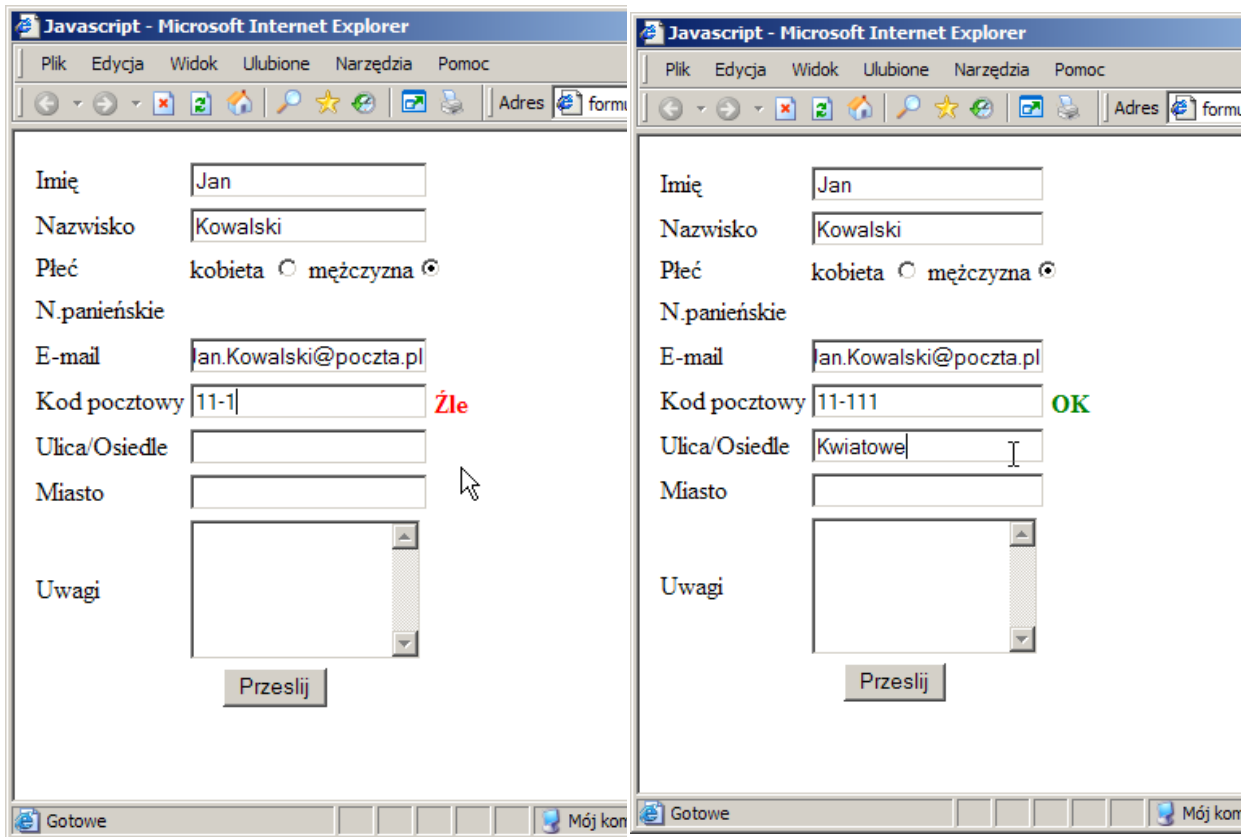
30. Konieczne jest zdefiniowanie funkcji sprawdzającej wpisany tekst po każdym puszczaniu klawisza. Dodaj funkcję `checkZIPCodeRegEx()` do pliku ze skryptami. Kod tej funkcji znajduje się poniżej.

```
function checkZIPCodeRegEx(e)  
{  
  var ZIPcode = /^[0-9]{2}-[0-9]{3}$/;  
  if (ZIPcode.test(e.value) )  
  {  
    document.getElementById("kod").innerHTML = "OK";  
    document.getElementById("kod").className = "green";  
    return true;  
  }  
  else  
  {  
    document.getElementById("kod").innerHTML = "Źle";  
    document.getElementById("kod").className = "red";  
    return false;  
  }  
}
```

31. W sekcji nagłówkowej dokumentu `formularz.html` dodaj do arkusza stylistycznego dodatkowe dwie reguły CSS

```
<style type="text/css">  
  span.err { color : red; font-weight : bold; padding-left : 5px }  
  .green   { color : green }  
  .red     { color : red   }  
</style>
```

32. W ciele funkcji `validate()` (plik `form_check.js`) zamień wywołanie funkcji `checkString(form.elements["f_kod"].value, ' Błędny kod pocztowy')` na `checkZIPCodeRegex(form.elements["f_kod"])`. Sprawdź działanie formularza.



33. JavaScript umożliwia łatwe iterowanie po elementach formularza. W sekcji nagłówkowej dokumentu `formularz.html` dodaj do arkusza stylistycznego dodatkowe dwie reguły CSS

```
<style type="text/css">
    span.err    { color : red; font-weight : bold; padding-left : 5px }
    .green      { color : green }
    .red        { color : red }
    .correct    { border-style: dotted; border-color: green }
    .wrong      { border-style: dotted; border-color: red }
</style>
```

Następnie, dodaj do pliku `form_check.js` poniższą funkcję:

```
function validate2(form)
{
    if (validate(form))
        for (i=0; i<form.elements.length; i++) {
            form.elements[i].setAttribute("class","correct");
        }
    else
        for (i=0; i<form.elements.length; i++) {
            form.elements[i].setAttribute("class","wrong");
        }
}
```

W ostatnim kroku w pliku formularz.html podmień wywołanie funkcji `validate(form)` w momencie kliknięcia przycisku **Przeslij** na wywołanie funkcji `validate2(form)`. Przetestuj wprowadzone zmiany.

34. W dotychczasowych przykładach wyszukiwaliśmy elementy według ich identyfikatorów (`getElementById`). Często jednak nie mamy takiej możliwości (np. dostęp do wierszy tabeli tworzonej w pętli). W takich przypadkach mogą się przydać metody przechodzenia po drzewie dokumentu. Wstaw poniższą tabelkę nad formularzem.

```
<table><tbody>
  <tr><td>Wiersz 1</td></tr>
  <tr><td>Wiersz 2</td></tr>
  <tr><td>Wiersz 3</td></tr>
  <tr><td>Wiersz 4</td></tr>
  <tr><td>Wiersz 5</td></tr>
  <tr><td>Wiersz 6</td></tr>
  <tr><td>Wiersz 7</td></tr>
</tbody></table>
<form name="dane_osobowe">
...
```

Aby uatrakcyjnić jej wygląd pokolorujemy co drugi wiersz. Dodaj do pliku ze skryptami poniższy kod,

```
function alterRows(i, e) {
  if (e) {
    if (i % 2 == 1)
      e.setAttribute("style", "background-color: Aqua;");
    e = e.nextSibling;
    while (e && e.nodeType != 1)
      e = e.nextSibling;
    alterRows(++i, e);
  }
}
```

a następnie w pliku formularz.html dodaj w sekcji script jej wywołanie.

```
<script type="text/javascript">
  var sib = document.getElementsByTagName("tr")[0];
  alterRows(1, sib);
...
```

Posłużyliśmy się metodą `getElementsByTagName`, która wyszukuje wszystkie elementy o zadanej etykiecie. Metoda ta działa w kontekście elementu na którym jest wywołana, przeszukując wszystkie węzły drzewa znajdujące się w poniżej. Następnie wykorzystujemy metodę `nextSibling`, która zwraca następny węzeł będący rodzeństwem aktualnego. Innymi metodami pozwalającymi na przechodzenie po węzłach są `firstChild`, `lastChild`, `previousSibling`, `parentNode` oraz `childNodes`. Należy również mieć na uwadze, iż w drzewie dokumentu węzły to nie tylko elementy, ale też na przykład komentarze czy tekst! Informację o typie węzła przechowuje pole `nodeType`, którego wartość 1 oznacza element.

35. W tym przykładzie pokażemy w jaki sposób można manipulować zawartością strony przy użyciu języka JavaScript. Wstaw poniższe funkcje do pliku ze skryptami.

```
function nextNode(e) {
    while (e && e.nodeType != 1)
        e = e.nextSibling;
    return e;
}

function prevNode(e) {
    while (e && e.nodeType != 1)
        e = e.previousSibling;
    return e;
}

function swapRows(b) {
    var tab = prevNode(b.previousSibling);
    var tBody = nextNode(tab.firstChild);
    var lastNode = prevNode(tBody.lastChild);
    tBody.removeChild(lastNode);
    var firstNode = nextNode(tBody.firstChild);
    tBody.insertBefore(lastNode, firstNode);
}
```

Dwie pierwsze funkcje zwracają najbliższy element w przód lub w tył. Kolejna funkcja posłuży nam aby wstawić ostatni wiersz tabeli jako pierwszy. Zauważ, że mimo iż usuwamy element z dokumentu nie jest on niszczone i można ponownie wstawić go do dokumentu. Wstaw do dokumentu pod tabelką a przed formularzem przycisk, który wywoła metodę zamiany. Zauważ, że metoda zadziała tylko wtedy, gdy element wywołujący zdarzenie będzie znajdował się bezpośrednio za tabelą.

```
...
</table>
<button onclick="swapRows(this)">Swap</button>
<form name="dane_osobowe">
...
```

36. Kolejny przykład ilustruje wykorzystanie języka JavaScript do kontroli długości wprowadzanego tekstu. W pliku formularz.html wprowadź, bezpośrednio po polu typu memo, obszar do wyświetlania licznika znaków:

```
<tr>
  <td>Uwagi</td>
  <td>
    <textarea rows="5" cols="15" name="uwagi"
      onKeyDown = "cnt( this.form.uwagi, document.getElementById('zostalo'), 150)">
    </textarea></td>
</tr>
<tr>
  <td>Pozostało <span id="zostalo">150</span> znaków</td>
</tr>
...
```

Dodaj do pliku form_check.js kod funkcji cnt():

```
function cnt(form, msg, maxSize) {
    if (form.value.length > maxSize)
        form.value = form.value.substring(0,maxSize);
    else
        msg.innerHTML = maxSize - form.value.length;
}
```

Przetestuj wprowadzoną funkcjonalność.