Spis treści

1 Szkielet strony pod formularz 2
2 Szkielet formularza i sposoby jego wysyłania 2
3 Kontrolki formularza 4
3.1 Przycisk wysłania formularza - submit 4
3.2 Etykieta - Label 5
3.3 Pole tekstowe - text 5
3.4 Wielowierszowe pole tekstowe - textarea 6
3.5 Pole do wpisywania hasła - password 6
3.6 Pole wielokrotnego wyboru - checkbox 7
3.7 Pole jednokrotnego wyboru - radio 8
3.8 Grupowanie pól 9
3.9 Pole listy (listbox) i listy rozwijalnej (combobox) - select 10
3.10 Pole ukryte - hidden 14
3.11 Graficzny przycisk wysyłania formularza - image 14
3.12 Przycisk czyszczenia formularza - reset 15
3.13 Przyciski ogólnego zastosowania 15
3.14 Wysyłanie plików na serwer - file 16

1 Szkielet strony pod formularz

```
<!DOCTYPE html>
<html lang="pl">

<head>
      <meta charset="UTF-8">
</head>

<body>
</body>

</html>
```

Treść strony, formularz umieszczamy między znacznik `body`.

2 Szkielet formularza i sposoby jego wysyłania

Podstawowe ramy formularza wyznacza znacznik `<form>`. Znacznik ten posiada kilka atrybutów których można użyć. Jednym z nich jest atrybut `action`. Jest to atrybut obowiązkowy, który określa gdzie mają zostać wysłane dane z formularza. Jego wartością może być dowolny adres URL. Zazwyczaj wstawia się tu adres rozpoczynający się od `http://` lub `https://`, wymaga to jednak posiadania skryptu który uruchomi się na serwerze i przetworzy przesłane dane (np. skrypt napisany w PHP). Jeżeli nie masz możliwości skorzystania z takiego skryptu, możesz skorzystać z możliwości wysłania zawartości formularza na adres email, podając swój adres email w formacie `mailto:ktos@gdzies.com.pl`.

Drugim atrybutem który warto zawsze ustawiać, jest `method`. Może on przyjąć dwie wartości:

- `get` Dane jakie będziemy chcieli przekazać do kolejnej strony będą znajdowały się w adresie URL np. Jeśli zmienną `gora` wyślemy do strony `sudety.php` a wartość ustawimy na `1422m` to po zatwierdzeniu przyciskiem formularza ukarze nam się strona o adresie: `http://localhost/sudety.php?gora=1422m`
  Serwery UNIX ograniczają długość takiego URL do 1240 bajtów.

- `post` Informacje przekazywane są ze strony na stronę w zmiennych zadeklarowanych w odpowiednich polach formularza.

W przypadku gdy chcesz wysłać dane z formularza na swój adres email, użyj `post` (`get` nie działa poprawnie w takim wypadku - w emailu nic nie będzie). Jeżeli natomiast dane chcesz wysłać do skryptu na serwerze, możesz użyć dowolnej z tych dwóch wartości. Zazwyczaj jednak stosuje się też `post`. Zatem nasz szkielet formularza którego zawartość można wysłać na email będzie wyglądał następująco:

```
<form action="mailto:ktos@gdzies.com.pl" method="post">
      Tutaj będzie zawartość formularza
</form>
```

Lub

```
<form action="mailto:adres e-mail?subject=temat" method="post">
        (Tutaj umieszcza się pola formularza)
</form>
```

Warto wspomnieć o tym iż warunkiem zgodności kodu HTML strony ze standardem (X)HTML Strict jest aby zawartość formularza umieścić wewnątrz znacznika będącego pojemnikiem, takim jak np. `<div>`. Uwzględniając to zalecenie szkielet formularza będzie wyglądał następująco:

```
<form action="mailto:ktos@gdzies.com.pl" method="post">
      <div>
      Tutaj będzie zawartość formularza
      </div>
</form>
```

Zawartość formularza jest przed wysyłaniem odpowiednio przetwarzana. Domyślnie dane z formularza kodowane są wg standardu URL Encode, przez co zawartość emaila jest raczej mało czytelna:

```
imie=Daniel&nazwisko=Fru%BFy%F1ski&strona=http%3A%2F%2Fwww.poradnik-
webmastera.com
```

Powyższy ciąg znaków jest to wynik wysłania formularza który zawierał pola `imie`, `nazwisko` i `adres`, do których wpisałem swoje imię, nazwisko i adres strony WWW. Jak widzisz, polskie znaki oraz niektóre znaki nie będące literami lub cyframi zostały zakodowane w postaci `%XX`, co jest mało czytelne (dla dociekliwych: XX jest to kod znaku w systemie szesnastkowym). Nazwy pól są połączone z ich wartościami za pomocą znaków "=", a wszystkie takie pary są dodatkowo połączone w jeden długi ciąg znaków za pomocą znaków "&".

Sposób kodowania danych z formularza można określić za pomocą dodatkowego atrybutu `enctype`. Aby zawartość wysyłanego maila była czytelna, należy mu przypisać wartość `text/plain`, co w wolnym tłumaczeniu oznacza "czysty tekst". Po jego dodaniu, szkielet formularza będzie wyglądał następująco:

```
<form action="mailto:ktos@gdzies.com.pl" method="post" enctype="text/plain">
      <div>
            Tutaj będzie zawartość formularza
      </div>
</form>
```

Zawartość tak przygotowanego i wysłanego formularza powinna wyglądać następująco:

```
imie=Daniel
nazwisko=Frużyński
strona=http://www.poradnik-webmastera.com
```

Niestety u mnie zamiast polskich znaków pojawiły się krzaczki, których nie udało mi się pozbyć. Testowałem też różne kombinacje czterech przeglądarek WWW (Firefox, Opera, Netscape Browser i IE) i dwóch klientów poczty (The Bat! i Outlook Express), i wszędzie dostawałem podobne rezultaty. Zatem albo u mnie coś jest nie tak, albo to już tak działa :)

3 Kontrolki formularza

Oczywiście każdy formularz poza "szkieletem" powinien coś w sobie zawierać. Poza tekstem który jest zazwyczaj używany do opisania pól formularza, można umieścić kontrolki do wprowadzania tekstu, przyciski opcji (radiowe), przyciski wyboru (checkbox), listy, listy rozwijalne i przyciski (zwykłe i graficzne).

3.1 Przycisk wysłania formularza - submit

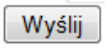
```
<input type="hidden" name="id" value="123">
```

Po wypełnieniu formularza, użytkownik będzie chciał go wysłać. Aby mu to umożliwić, należy użyć znacznika:

```
<input type="submit">
```

Znacznik ten może być użyty bez atrybutów `name` i `value`. W przypadku gdy nie poda się pierwszego z nich, przeglądarka nie dołączy tego elementu do listy par `nazwa=wartość` które zostaną wysłane. Gdy natomiast nie podasz wartości dla atrybutu `value`, przeglądarka użyje swojego domyślnego opisu przycisku (np. IE wyświetla enigmatycznie brzmiący napis "Prześlij kwerendę").

```
<form action="obsluga.php" method="post">
     <input type="submit" name="wyslij" value="Wyślij">
</form>
```



W pliku `obsluga.php` należy umieścić kod obsługujący formularz.

W pliku .php odczytujmy dane z formularza poprzez atrybut `name` kontrolek.
Sposoby, na które skrypt php może zaakceptować dane od użytkownika:

- `$imie` styl prosty, krótki

  Dyrektywa: `register_globals` w pliku php.ini
  Metoda używana w celu tworzenia zmiennych do przechowywania informacji z formularzy sieciowych, zależy od wersji PHP. Poniżej wersji 4.2.0 dyrektywa ta jest ustawiona na „on” pozwalając na styl krótki.
  Może wystąpić problem kolizji nazw z formularza z tymi umieszczonymi np. w ciasteczkach, sesjach, itp. W nowszych wersjach PHP dyrektywa ta jest ustawiona na „off”. Zalecany jest styl średni.

- **`$_POST['imie']`** styl średni, można użyć `$_GET['imie']` w zależności jaką metodą (`metod`) przesyłamy dane.

- `$_HTTP_POST_VARS['imie'], $_HTTP_GET_VARS['imie']` styl długi – niestosowany.

Znacznik `<input>` jest używany także do tworzenia wielu innych elementów formularza - zależy to od wartości atrybutu `type`.

3.2 Etykieta - Label

Dla wszystkich pól formularza (z wyjątkiem przycisków i pól ukrytych) powinno się dodawać **etykiety** nazywające pole, których kliknięcie spowoduje uaktywnienie określonej kontrolki. Można to zrobić na dwa sposoby:

a) poprzez umieszczenie pola wewnątrz elementu `<label>`

```
<label>Imię <input type="text" name="imie[]"/></label>
```

b) poprzez nadanie *id* polu (dowolnego, może być różne od *name*), każda kontrolka powinna mieć inny identyfikator, nawet jeżeli mają te same nazwy nadane atrybutem *name*. Następnie identyfikatory te użyj jako wartości atrybutu *for* znacznika `<label>`. W połączeniu z CSSowym `inline-block` pozwala nadać szerokość etykietom i tym samym wizualnie wyrównać pola formularza

```
<label for="imie1">Imię</label> <input type="text" name="imie[]"
id="imie1"/>
```

Ten element jest szczególnie ważny dla pól typu checkbox i radio, bo powoduje, że kliknięcie etykiety zaznacza pole (tak, jak to ma miejsce w większości systemów operacyjnych).

```
<input type="radio" name="odpowiedz" id="odp_tak" value="tak"><label
for="odp_tak">Tak</label><br>

<input type="radio" name="odpowiedz" id="odp_nie" value="nie"><label
for="odp_nie">Nie</label><br>

<input type="radio" name="odpowiedz" id="odp_niewiem" value="niewiem"><label
for="odp_niewiem">Nie wiem</label>
```

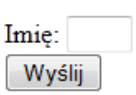
3.3 Pole tekstowe - text

Atrybuty:

`name`	Nazwa pola, powinna być unikatowa w obrębie formularza (unikamy polskich znaków oraz spacji)
`value`	Wartość początkowa pola
`size`	Rozmiar pola w znakach
`maxlength`	Maksymalny rozmiar pola
`tabindex="n"`	Kolejność przy przenoszeniu między polami po naciśnięciu klawisza TAB. gdzie "n" to liczba z przedziału od 1 do 32767

HTML:

```
<form action="obsluga.php" method="post">
      Imię: <input type="text" name="imie" size="3" maxlength="15">
      <br>
      <input type=submit name="wyslij" value="Wyślij">
</form>
```



PHP: plik `obsluga.php`

```
<?php
echo $_POST['imie'];
?>
```

3.4 Wielowierszowe pole tekstowe - textarea

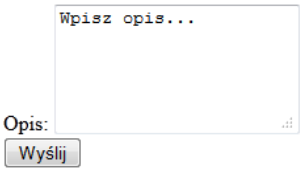
Atrybuty:

`name`	Nazwa pola, powinna być unikatowa w obrębie formularza (unikamy polskich znaków oraz spacji)
`cols="x" rows="y"`	"x" (ilość kolumn) oraz "y" (ilość wierszy) tekstu, które mogą się jednocześnie zmieścić w polu, bez jego przewijania.

Objętość pola do 1024 znaków.
Znacznik ten jest nieco inny od znacznika `<input>` - nie wymaga atrybutu `type`.
Dodatkowo istnieje różnica w sposobie podawania tekstu który ma zostać domyślnie wyświetlony - wpisuje się go pomiędzy znacznik otwierający a zamykający.

HTML:

```
<form action="obsluga.php" method="post">
  Opis: <textarea name="opis" cols="20" rows="5">Wpisz opis...</textarea>
  <br>
  <input type=submit name="wyslij" value="Wyślij">
</form>
```



PHP: plik `obsluga.php`

```
<?php
echo $_POST['opis'];
?>
```

3.5 Pole do wpisywania hasła - password

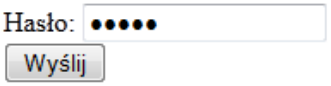
Atrybuty:

`name`	Nazwa pola, powinna być unikatowa w obrębie formularza (unikamy polskich znaków oraz spacji)
`value`	Wartość początkowa pola
`size`	Rozmiar pola w znakach
`maxlength`	Maksymalny rozmiar pola
`tabindex="n"`	Kolejność przy przenoszeniu między polami po naciśnięciu klawisza TAB. gdzie "n" to liczba z przedziału od 1 do 32767

Wprowadzany tekst jest urywany pod gwiazdkami.

HTML:

```
<form action="obsluga.php" method="post">
      Hasło: <input type="password" name="haslo">
      <br>
      <input type=submit name="wyslij" value="Wyślij">
</form>
```



PHP: plik `obsluga.php`

```
<?php
echo $_POST['haslo'];
?>
```

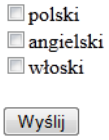
3.6 Pole wielokrotnego wyboru - checkbox

Atrybuty:

`name`	Nazwa pola, powinna być unikatowa w obrębie formularza (unikamy polskich znaków oraz spacji). Każde kolejne pole zawiera tę samą nazwę, gdyż pytamy o ten sam rodzaj danych (tę samą zmienną).
`value`	Wartość zmieniająca się wraz z kolejnym wariantem.
`checked`	Domyślne zaznaczenie pola (patrz opis poniżej)
`disabled`	Zablokowanie pola (patrz opis poniżej)

```
<!-- Default to unchecked -->
<input type="checkbox">

<!-- Default to checked, XHTML -->
<input type="checkbox" checked="checked" />

<!-- Default to checked, HTML5 -->
<input type="checkbox" checked>
```

Pole ma kształt kwadratu.



HTML:

```
<form action="obsluga.php" method="post">
  <input type="checkbox" name="jezyk[]" value="polski">polski
  <br>
  <input type="checkbox" name="jezyk[]" value="angielski">angielski
  <br>
  <input type="checkbox" name="jezyk[]" value="włoski">włoski
  <br>
  <br>
  <input type=submit name="wyslij" value="Wyślij">
</form>
```

PHP: plik `obsluga.php`

```
<?php

if( isset($_POST['jezyk']) )  // Jeżeli jest coś włączone...
 for( $i = 0; $i < count($_POST['jezyk']); $i++ )
 {
       echo 'Checked: '.$_POST['jezyk'][$i].'<br>';
 };


/*
if( isset($_POST['jezyk']) )  // Jeżeli jest coś właczone...
foreach($_POST['jezyk'] as $value)
{
      if( isset($value) )
       echo 'Checked: '.$value.'<br>';
};
*/
?>
```

3.7 Pole jednokrotnego wyboru - radio

Atrybuty:

`name`	Nazwa pola, powinna być unikatowa w obrębie formularza (unikamy polskich znaków oraz spacji). Każde kolejne pole zawiera tę samą nazwę, gdyż pytamy o ten sam rodzaj danych (tę samą zmienną).
`value`	Wartość zmieniająca się wraz z kolejnym wariantem.
`checked`	Domyślne zaznaczenie pola (patrz opis poniżej)
`disabled`	Zablokowanie pola (patrz opis poniżej)

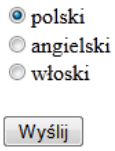
```
<!-- Default to unchecked -->
<input type="checkbox">

<!-- Default to checked, XHTML -->
<input type="checkbox" checked="checked" />

<!-- Default to checked, HTML5 -->
<input type="checkbox" checked>
```

Pole jest okrągłe.

Wszystkie pola tego typu, które dotyczą tego samego pytania (należą do tej samej grupy), koniecznie muszą mieć taką samą nazwę (atrybut *name*). Dodatkowo atrybut `value="..."` jest tutaj absolutnie konieczny - musi on być inny dla każdej możliwości odpowiedzi.

1) Tablica

HTML:




```
<form action="obsluga.php" method="post">
  <input type="radio" name="jezyk[]" value="polski">polski
  <br>
  <input type="radio" name="jezyk[]" value="angielski">angielski
  <br>
  <input type="radio" name="jezyk[]" value="włoski">włoski
  <br>
  <br>
  <input type=submit name="wyslij" value="Wyślij">
</form>
```

PHP: plik `obsluga.php`

```
<?php

if( isset($_POST['jezyk']) )  // Jeżeli jest coś włączone...
 for( $i = 0; $i < count($_POST['jezyk']); $i++ )
 {
       echo 'Checked: '.$_POST['jezyk'][$i].'<br>';
 };

/*
if( isset($_POST['jezyk']) )  // Jeżeli jest coś właczone...
foreach($_POST['jezyk'] as $value)
{
      if( isset($value) )
       echo 'Checked: '.$value.'<br>';
};
*/
?>
```

2)
HTML:

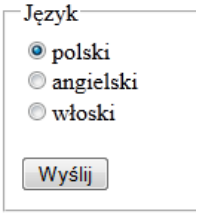
```
<form action="obsluga.php" method="post">
  <input type="radio" name="jezyk" value="polski" checked>polski
  <br>
  <input type="radio" name="jezyk" value="angielski">angielski
  <br>
  <input type="radio" name="jezyk" value="włoski">włoski
  <br>
  <br>
  <input type=submit name="wyslij" value="Wyślij">
</form>
```

PHP: plik `obsluga.php`

```
<?php
      echo 'Checked: '.$_POST['jezyk'];
?>
```

3.8 Grupowanie pól

Ramka otaczająca tekst lub grupę pól.



Wersja podstawowa:

```
<fieldset>
      (Pola formularza)
</fieldset>
```

Wersja z tytułem:

```
<fieldset>
<legend>Tytuł</legend>
      (Pola formularza)
</fieldset>
```

Ułożenie tytułu:

```
<fieldset>
<legend align="rodzaj">Tytuł</legend>
      (Pola formularza)
</fieldset>
```

gdzie jako "*rodzaj*" należy wpisać:

`"left"`	tytuł położony przy lewej krawędzi ramki grupującej (domyślnie)
`"center"`	tytuł ułożony na środku
`"right"`	po prawej

3.9 Pole listy (listbox) i listy rozwijalnej (combobox) - select

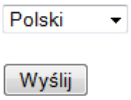
Do formularza można także wstawić pola listy (`listbox`) i pola listy rozwijalnej (`combobox`). Obydwa można utworzyć za pomocą znacznika `<select>`. Znacznik ten posiada atrybut `size`, który określa ile elementów ma być widocznych, czyli inaczej wysokość listy. Jeżeli ustawisz go na 1, to otrzymasz pole listy rozwijalnej; jeżeli natomiast użyjesz wartości większej od 1, otrzymasz zwykłe pole listy.

Wszystkie elementy listy umieszcza się wewnątrz znacznika `<select>`, ograniczając je dodatkowo znacznikami `<option>`.

Przykład listy rozwijalnej (combobox):

HTML:

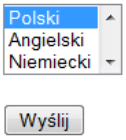
```
<select name="jezyk" size="1">
	<option>Polski</option>
	<option>Angielski</option>
	<option>Niemiecki</option>
	<option>Francuski</option>
</select>
```



Przykład pola listy (listbox):

HTML:

```
<select name="jezyk" size="3">
	<option>Polski</option>
	<option>Angielski</option>
	<option>Niemiecki</option>
	<option>Francuski</option>
</select>
```



PHP: plik `obsluga.php`

```
<?php
	echo 'Wybrano: '.$_POST['jezyk'];
?>
```

Na liście rozwijalnej (combo) domyślnie wybrany jest pierwszy element. Jeżeli chcesz aby domyślnie nic nie było wybrane, dodaj tam na początku pusty element.

Wartością która zostanie wysłana do serwera po wybraniu któregoś z elementów jest zawartość znacznika `<option> </option>`. Jeżeli chcesz aby były przesyłane inne wartości, możesz je podać za pomocą atrybutu `value` (przykład dla listbox i dla combobox jest taki sam):

HTML:

```
<select name="jezyk" size="3">
	<option value="pl">Polski</option>
	<option value="en">Angielski</option>
	<option value="de">Niemiecki</option>
	<option value="fr">Francuski</option>
</select>
```

PHP: plik `obsluga.php`

```
<?php
	//echo 'Wybrano: '.$_POST['jezyk'];

	switch ($_POST['jezyk'])
	{
		case 'pl':
			echo 'Polski!';
		break;

		case 'en':
			echo 'Angielski!';
		break;

		case 'de':
			echo 'Niemiecki!';
		break;

		case 'fr':
			echo 'Francuski!';
		break;
	}
?>
```

Możliwość wybrania kilku opcji

W polu listy (listbox) można domyślnie zaznaczyć tylko jeden element. Możesz to zmienić poprzez dodanie do znacznika `<select>` atrybutu `multiple`. Atrybut ten nie wymaga podania żadnej wartości. Po jego użyciu, będzie możliwe zaznaczenie większej ilości elementów poprzez klikanie na nich z naciśniętym klawiszem `Ctrl` na klawiaturze.

Dla każdej z list (listbox i combobox) można określić który element ma być domyślnie wybrany. Służy do tego atrybut `selected`, który należy umieścić we właściwym znaczniku `<option>`. Nie wymaga on podawania wartości. W przypadku gdy używać pola listy (listbox) i atrybutu `multiple`, możesz użyć tego atrybutu kilkakrotnie aby zaznaczyć kilka elementów listy.

Pole można zablokować atrybutem `disabled`.

Aby odebrać wszystkie zaznaczone opcje w skrypcie PHP, jako nazwę pola można wpisać: `name="nazwa[]"`. W takim przypadku w skrypcie będzie dostępna tablica `$nazwa` (`$_POST['nazwa']` lub `$_GET['nazwa']`), a wartości kolejnych zaznaczonych opcji (tylko zaznaczonych), będzie można odczytać poprzez użycie:

a) `$nazwa[0]`, `$nazwa[1]`, `$nazwa[2]`, …
b) `$_POST['nazwa'][0]`, `$_POST['nazwa'][1]`, `$_POST['nazwa'][2]` albo `$_GET['nazwa'][0]`, `$_GET['nazwa'][1]`, `$_GET['nazwa'][2]`) itd. (w zależności ile opcji zostanie zaznaczonych).

HTML:

```
<form action="obsluga.php" method="post">
     <select name="jezyk[]" size="4" multiple>
          <option value="pl" selected>Polski</option>
          <option value="en">Angielski</option>
          <option value="de">Niemiecki</option>
          <option value="fr">Francuski</option>
     </select>
     <br>
     <br>
     <input type=submit name="wyslij" value="Wyślij">
</form>
```

PHP: plik `obsluga.php`

```
<?php
 if( isset($_POST['jezyk']) )    // Jeżeli jest coś włączone...
  for( $i = 0; $i < count($_POST['jezyk']); $i++ )
  {
       echo 'Wybrano: '.$_POST['jezyk'][$i].'<br>';
  };
?>
```

Wybrano: pl
Wybrano: de

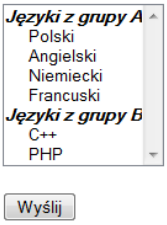
Lub

```
<?php
 if( isset($_POST['jezyk']) )    // Jeżeli jest coś włączone...
  foreach($_POST['jezyk'] as $value)
  {
     if( isset($value) )
      echo 'Wybrano: '.$value.'<br>';
  };
?>
```

Grupy opcji

Czasem zachodzi potrzeba wizualnego zgrupowania kilku opcji listy rozwijalnej. Można oczywiście próbować wstawiać pomiędzy kolejne grupy znaczniki OPTION zawierające np. kilka znaków myślnika lub po prostu pustą zawartość. Niestety opcje takie tak naprawdę niczym nie będą różniły się od pozostałych, tzn. będzie je można normalnie wybrać. Można jednak dokonać grupowania za pomocą osobnego znacznika `OPTGROUP`. Pojawi się on w postaci nagłówka nad opcjami, które zawiera. Będzie wyróżniony wizualnie i nie będzie możliwe jego wybranie. Jedna lista rozwijalna może zawierać dowolną liczbę grup opcji, ale nie można ich zagnieżdżać, tzn. umieszczać jednej grupy wewnątrz innej.

HTML:




```
<form action="obsluga.php" method="post">
     <select name="jezyk[]" size="8" multiple>
          <optgroup label="Języki z grupy A">
               <option value="pl">Polski</option>
               <option value="en">Angielski</option>
               <option value="de">Niemiecki</option>
               <option value="fr">Francuski</option>
          </optgroup>

          <optgroup label="Języki z grupy B">
               <option value="cpp">C++</option>
               <option value="php">PHP</option>
          </optgroup>
     </select>
     <br>
     <br>
     <input type=submit name="wyslij" value="Wyślij">
```

PHP: plik `obsluga.php`

```
<?php
/*
if( isset($_POST['jezyk']) )     // Jeżeli jest coś włączone...
 for( $i = 0; $i < count($_POST['jezyk']); $i++ )
 {
      echo 'Wybrano: '.$_POST['jezyk'][$i].'<br>';
 };
*/

if( isset($_POST['jezyk']) )     // Jeżeli jest coś włączone...
 foreach($_POST['jezyk'] as $value)
 {
     if( isset($value) )
      echo 'Wybrano: '.$value.'<br>';
 };
?>
```

Przeglądarki nieco odmiennie wyświetlają listy wyboru z grupowaniem opcji - w IE7 i Firefoksie tytuły bloków opcji są pogrubione i pochylone, a w Operze białe na czarnym tle. Sposób wyświetlania możemy samodzielnie zmienić, nadając poleceniu `optgroup` odpowiednie style, np. czerwony kolor czy niebieskie tło.

3.10 Pole ukryte - hidden

Polecenie takie tworzy ukryte pole w formularzu. Nie jest ono widoczne ani dostępne dla użytkownika, ale jego wartość (`value="wartość"`) jest przesyłana wraz z formularzem. Może ono służyć np. dla podania informacji o numerze wersji ankiety, dacie jej ostatniej aktualizacji, adresu strony, z której została wysłana, opisu całego formularza lub też poszczególnych jego pól.

Zawartość tego znacznika może każdy podejrzeć (a nawet jak się postara, to też zmodyfikuje w dowolny sposób), więc ze względów bezpieczeństwa polecam nie wstawiać tutaj byle czego, i sprawdzać to co zostało wysłane do serwera (ta uwaga dotyczy wszystkich danych które pochodzą "z zewnątrz").

Dla zaawansowanych

Ukryte dane są często wykorzystywane do przekazywania informacji pomiędzy klientem a serwerem (skrypty po stronie serwera), które w przeciwnym razie zostałyby utracone, z uwagi na bezstanowy charakter protokołu HTTP.

HTML:

```
<form action="obsluga.php" method="post">
  <input type="hidden" name="ukryty" value="Ukryta treść" />
  <br>
  <br>
  <input type=submit name="wyslij" value="Wyślij">
</form>
```

PHP: plik `obsluga.php`

```
<?php
  echo $_POST['ukryty'];
?>
```

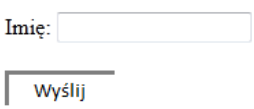
3.11 Graficzny przycisk wysyłania formularza - image

Drugą metodą wysyłania formularza jest przycisk graficzny. Tworzy się go za pomocą znacznika `<input type="image">`. Znacznik ten jest hybrydą znacznika `<input>` i znacznika `<img>`.

```
<input type="image" src="webdir.gif" alt="Wyślij">
```

Warto zauważyć że po kliknięciu na taki obrazek przeglądarka wyśle także współrzędne kliknięcia, w postaci dodatkowych parametrów x i y. Jeżeli do tego przycisku graficznego przypiszesz jakąś nazwę za pomocą atrybutu `name` (np. *wyslij*), to przeglądarka wyśle współrzędne jako parametry `wyslij_x` i `wyslij_y`.

HTML:




```
<form action="obsluga.php" method="post">
  Imię: <input type="text" name="imie">
  <br>
  <br>
  <input type="image" src="wyslij.png" name="wyslij" alt="Wyślij">
</form>
```

PHP: plik `obsluga.php`

```
<?php
  echo $_POST['imie']."<br>";
  echo 'X = '.$_POST['wyslij_x']."<br>";
  echo 'Y = '.$_POST['wyslij_y'];
?>
```

Tomasz
X = 47
Y = 11

Trzecią metodą wysyłania formularza jest naciśnięcie klawisza Enter na klawiaturze w momencie gdy aktywne jest któreś z pól formularza. Formularz nie musi mieć przycisku `<input type="submit">` aby ta metoda zadziałała.

3.12 Przycisk czyszczenia formularza - reset

Jego naciśnięcie spowoduje usunięcie wszystkich wartości wpisanych do formularza i zastąpienie ich wartościami domyślnymi (czyli takimi które pojawiły się zaraz po załadowaniu strony).

Podobnie jak przycisk `<input type="submit">`, ten przycisk może nie mieć podanej nazwy i/lub wartości. Podanie nazwy zazwyczaj jest zbędne, gdyż przycisk ten nie powoduje wysłania formularza. Częściej stosuje się podawanie wartości, gdyż jest ona wyświetlana jako tekst na przycisku (jeżeli się jej nie poda, wyświetli się wartość domyślna zależna od przeglądarki, np. "Zresetuj", co może być mało oczywiste).

HTML:

```
<form action="obsluga.php" method="post">
      Imię: <input type="text" name="imie">
      <br>
      <br>
      <input type="reset" value="Wyczyść formularz">
      <br>
      <input type="submit" name="wyslij" value="Wyślij">
</form>
```

Imię:
Wyczyść formularz
Wyślij

3.13 Przyciski ogólnego zastosowania

Oprócz wymienionych przycisków *submit* i *reset*, możliwe jest także wstawienie "zwykłych" przycisków. Można je utworzyć na dwa sposoby:

```
<input type="rodzaj" value="Przycisk 1"><br>
```

Przycisk 1
Przycisk 2

gdzie "rodzaj" określa typ przycisku i można tutaj podać:

`"submit"`	przycisk wysłania formularza
`"reset"`	przycisk wyczyszczenia danych
`"button"`	zwykły przycisk (domyślnie)

Lub:

```
<button>Przycisk 2</button>
```

Atrybuty:

`disabled`	Zablokowanie przycisku.

W przypadku zastosowania `type="button"` na ekranie pojawi się przycisk, po kliknięciu którego, nie nastąpi żadna akcja. Aby to zmienić, należy przechwycić wygenerowane zdarzenie *onclick*. Natomiast Netscape 7 oraz Opera 6, 7 traktują `<button>...</button>` jak przycisk wysłania formularza.

Przycisk stworzony za pomocą znacznika `<button> </button>` pozwala umieścić na nim nie tylko tekst, ale też np. obrazek:

```
<button>
  <img src="webdir.gif" width="101" height="77" alt="Kliknij">
</button>
```

3.14 Wysyłanie plików na serwer - file

Pole INPUT typu *file* służy do przekazywania za pomocą formularza pliku na serwer. Kontrolka taka jest przedstawiana w formularzu w postaci przycisku z napisem `Przeglądaj...`. Po naciśnięciu przycisku, przeglądarka wyświetla okno dialogowe, które umożliwia wybór pliku. Po dokonaniu wyboru i zatwierdzeniu formularza, na serwer zostaje przekazany wybrany plik.

Formularz zawierający kontrolki typu *file* musi być przekazywany metodą POST.

Atrybuty:

`file` typ pola, w tym przypadku wartość powinna być: `file`
`name` ustala nazwę zmiennej przekazanej do skryptu:

Dodatkowo trzeba ustawić typ kodowania formularza na `multipart/form-data` za pomocą atrybutu `enctype` znacznika `<form>`:

```
<form action="..." enctype="multipart/form-data">
     <input type="file" name="nazwa" />
</form>
```

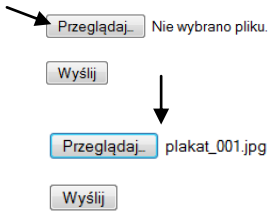
Po stronie serwera (PHP):
W jaki sposób odczytać zawartość plików przekazanych na serwer w języku php?

W odróżnieniu do innych kontrolek, pliki przekazane na serwer nie są zawarte w tablicy `$_POST`, a w tablicy **`$_FILES`**. Tablica `$_FILES` jest tablicą wielowymiarową. Jeśli kontrolka INPUT posiada atrybut `name="plik"`, wówczas informacje na temat pliku przekazanego na serwer są zawarte w następujących polach tablicy `$_FILES`:

`$_FILES['plik']['name']`	Oryginalna nazwa wysyłanego pliku.
`$_FILES['plik']['type']`	Typ MIME wysyłanego pliku (JPEG, GIF, ...).
`$_FILES['plik']['size']`	Rozmiar wysyłanego pliku (w bajtach)
`$_FILES['plik']['tmp_name']`	Tymczasowa nazwa pliku, który został wysłany na serwer.
`$_FILES['plik']['error']`	Numer błędu (0 oznacza prawidłowe wysłanie).

HTML:

```
<form action="obsluga.php" method="post" enctype="multipart/form-data">
     <input type="file" name="plik" />
     <br>
     <br>
     <input type="submit" name="wyslij" value="Wyślij">
</form>
```



PHP: plik `obsluga.php`

```
<?php
  echo $_FILES['plik']['name']."<br>";
  echo $_FILES['plik']['type']."<br>";
  echo $_FILES['plik']['size']."<br>";
  echo $_FILES['plik']['tmp_name']."<br>";
  echo $_FILES['plik']['error']."<br>";
?>
```

plakat_001.jpg
image/jpeg
21761
C:\Programy\Serwer\xampp\tmp\php3193.tmp
0

Tablica `$_FILES` zawiera informacje na temat przekazanego pliku, **nie zawiera samego pliku**. Plik jest zapisany na dysku pod nazwą `$_FILES['plik']['tmp_name']`. Jeśli chcemy przeczytać jego zawartość, to możemy do tego wykorzystać następującą instrukcję:

```
$rob = file_get_contents($_FILES['plik']['tmp_name']);
```

PHP: plik `obsluga.php`

```
<?php
  echo $_FILES['plik']['name']."<br>";
  echo $_FILES['plik']['type']."<br>";
  echo $_FILES['plik']['size']."<br>";
  echo $_FILES['plik']['tmp_name']."<br>";
  echo $_FILES['plik']['error']."<br>";
  echo "<br>";
  $rob = file_get_contents($_FILES['plik']['tmp_name']);
  echo $rob;
?>
```

plakat_001.jpg
image/jpeg
21761
C:\Programy\Serwer\xampp\tmp\php3BC1.tmp
0

JFIFExifII*Adobe Photosho
HLinomntrRGB XYZ 1acspMSFTIEC sRGB-H
textCopyright (c) 1998 Hewlett-Packard CompanydescsR(
http://www.iec.chIEC http://www.iec.chdesc.IEC 61966-2.

Należy pamiętać, że plik ten zostanie usunięty automatycznie po zakończeniu przetwarzania skryptu. Należy albo go przetworzyć, korzystając z powyższej instrukcji, albo przenieść do innego folderu stosując funkcję `move_uploaded_file()`. Przy przenoszeniu pliku należy wcześniej utworzyć folder do którego przerzucamy plik.

W naszym przykładzie strona jest w katalogu `formularz`

PHP: plik `obsluga.php`

```
<?php
      echo $_FILES['plik']['name']."<br>";
      echo $_FILES['plik']['type']."<br>";
      echo $_FILES['plik']['size']."<br>";
      echo $_FILES['plik']['tmp_name']."<br>";
      echo $_FILES['plik']['error']."<br>";
      echo "<br>";
      move_uploaded_file($_FILES['plik']['tmp_name'],
      $_SERVER['DOCUMENT_ROOT'].'/formularz/foto/'.$_FILES['plik']['name']);
?>
```

Plik zostanie przerzucony do katalogu `/formularz/foto`

W formularzu warto umieścić (przed kontrolką typu *file*) ukryte pole ustalające wartość zmiennej `MAX_FILE_SIZE`. Wartość taka ogranicza wielkość pliku, jaki możemy przekazać za pomocą formularza. Nie jest to zabezpieczenie przed złośliwymi użytkownikami, gdyż łatwo je ominąć (na przykład przygotowując własny formularz; do zabezpieczenia przed złośliwymi użytkownikami należy stosować opcje konfiguracyjne php: `file_uploads`, `upload_max_filesize` oraz `upload_tmp_dir`). Jednak dzięki zawarciu w formularzu informacji o maksymalnym rozmiarze pliku, unikniemy sytuacji, w której użytkownik czeka dłuższy odcinek czasu tylko po to, by zobaczyć komunikat (pochodzący ze skryptu php) informujący o tym, że wybrany plik jest zbyt duży. Jeśli informacja o maksymalnym rozmiarze jest zawarta w formularzu, to przeglądarka nie wysyła na serwer zbyt dużych plików.

```
<input type="hidden" name="MAX_FILE_SIZE" value="3000">
```
Podana wielkość jest mierzona w bajtach.

Przykład uploadera:
Źródło: `http://phpkurs.pl/upload/`

`upload.html:`
```
<form action="obsluga.php" method="post" enctype="multipart/form-data">
      <input type="file" name="plik" />
      <br>
      <br>
      <input type="submit" name="wyslij" value="Wyślij">
</form>
```

`obsluga.php:`
```
<?php
$max_rozmiar = 22*1024; // maksymalny rozmiar pliku 22kB

if (is_uploaded_file($_FILES['plik']['tmp_name']))
{
  if ($_FILES['plik']['size'] > $max_rozmiar)
  {
    echo 'Błąd! Plik jest za duży!';
  } else
  {
    echo 'Odebrano plik. Początkowa nazwa: '.$_FILES['plik']['name'];
    echo '<br/>';

    if (isset($_FILES['plik']['type']))
    {
      echo 'Typ: '.$_FILES['plik']['type'].'<br/>';
    }

    move_uploaded_file($_FILES['plik']['tmp_name'],
    $_SERVER['DOCUMENT_ROOT'].'/formularz/foto/'.$_FILES['plik']['name']);
  }
} else
{
  echo 'Błąd przy przesyłaniu danych!';
}
?>
```

Błąd przy przesyłaniu danych!

Odebrano plik. Początkowa nazwa: plakat_001.jpg
Typ: image/jpeg

Błąd! Plik jest za duży!

UWAGA:
Przesyłanie plików na serwer jest sprawą dosyć niebezpieczną, dlatego należy odpowiednio się zabezpieczyć. W powyższym przykładzie użyta została funkcja `is_uploaded_file()`. Sprawdza ona czy podany plik faktycznie został odebrany od użytkownika – sprawdzenie takie jest istotne, gdyż w przypadku źle napisanego skryptu "włamywacz" będzie mógł odczytać z serwera dowolny plik, do którego prawo odczytu posiada użytkownik jako który pracuje serwer WWW.

Po kolei wykorzystywane są dostępne informacje o pliku. Jeśli wszystkie próby przebiegną pomyślnie, plik jest przenoszony w docelowe miejsce przy pomocy funkcji `move_uploaded_file()`. Oczywiście jeśli plik o takiej samej nazwie już istnieje, zostanie nadpisany, dlatego też należy najpierw to sprawdzić przy pomocy funkcji `file_exists()`.

Kolejnym niebezpieczeństwem jest możliwość wstawienia na serwer skryptu PHP zawierającego "niebezpieczne instrukcje". Można się przeciw temu zabezpieczyć sprawdzając rozszerzenie lub typ przesyłanego pliku. Jeśli rozszerzenie pliku to `.php` (lub inne, które jest przetwarzane przez serwer WWW jako skrypt PHP) lub typ pliku jest inny od oczekiwanego (na przykład wszystkie inne niż `image/gif` czy `image/jpeg`), plik można albo usunąć albo zmienić mu rozszerzenie.

Aby plik mógł zostać przeniesiony w docelowe miejsce, docelowy katalog musi mieć odpowiednie prawa dostępu. Mianowicie użytkownik, jako który pracuje serwer WWW musi mieć prawo zapisu do tego katalogu. Wszystkie niezbędne informacje można uzyskać od administratora serwera, lub szukając dyrektywy `User` w pliku `/etc/httpd/httpd.conf`.

Źródło:
http://www.poradnik-webmastera.com/kursy/html/formularze.php