REFORMA 2012

Aplikacje internetowe

DO NOWEJ PODSTAWY PROGRAMOWEJ

Kwalifikacja E.14.3

Podręcznik do nauki zawodu

TECHNIK INFORMATYK

WSiP

REFORMA 2012

Aplikacje internetowe

Małgorzata Łokińska

Kwalifikacja E.14.3

Podręcznik do nauki zawodu

TECHNIK INFORMATYK

WSiP

I. JavaScript – skryptowy język programowania

II. PHP – obiektowy, skryptowy język programowania

I. JavaScript – skryptowy język programowania

- Umieszczanie skryptów w dokumencie
- Instrukcja dokument.write
- Okno dialogowe
- Typy danych
- Zmienne
- Operatory
- Instrukcje warunkowe
- Pętle
- Funkcje
- Obiekty
- Zdarzenia
- Obsługa formularzy

1 Umieszczanie skryptów w dokumencie

ZAGADNIENIA

- Czym jest JavaScript?
- Jakie są sposoby na umieszczenie skryptu JavaScript?
- Jak umieszczać skrypt w dokumencie HTML?
- Jak korzystać ze skryptów zewnętrznych?

JavaScript jest językiem skryptowym pozwalającym na rozszerzenie standardowych dokumentów HTML m.in. o możliwość interakcji z użytkownikiem oraz na sprawdzanie poprawności informacji wprowadzanych przez użytkowników. Powstał w połowie lat 90. XX wieku w firmie Netscape jako język skryptowy LiveScript. W 1995 roku w wyniku współpracy z firmą Sun Microsystems ugruntował się jako język JavaScript. Jego głównym autorem jest Brendan Eich.

Program napisany w języku programowania wysokiego poziomu (C, C++, JAVA) przed zastosowaniem **kodu źródłowego**, którym jest uporządkowany ciąg instrukcji, wymaga skompilowania. Procesem tym zajmuje się **kompilator**, program którego zadaniem jest przetłumaczenie całego kodu źródłowego na **kod maszynowy**, ciąg liczb w systemie dwójkowym zrozumiały dla procesora. Podobne zadanie wykonuje **interpreter**. Jego sposób działania opiera się na pobraniu pojedynczej linii kodu, przetłumaczeniu jej, przekazaniu do procesora i przejściu do kolejnej linii kodu. Proces ten powtarza się do czasu przetłumaczenia całego programu.

JavaScript jest interpretowanym językiem programowania. Żeby zobaczyć efekty działania programu, nie trzeba go kompilować do kodu maszynowego. Wystarczy przeglądarka internetowa mająca włączoną obsługę języka JavaScript.

Jest to jeden z popularniejszych języków programowania działających po stronie klienta. To jedna z największych jego zalet, ponieważ wszystkie przeprowadzane operacje nie obciążają serwera. Zapewnia on również odpowiedni poziom bezpieczeństwa, ponieważ nie ma dostępu do systemu plików znajdujących się na komputerze użytkownika. Nieco trudności sprawia jedynie jego interpretacja przez różne przeglądarki. Może się zdarzyć, że prawidłowo napisany program będzie różnie działał w zależności od obsługującej go przeglądarki.

Sam język jest łatwy do opanowania. Do stworzenia dowolnego programu wystarczy zwykły edytor tekstu (np. Notatnik) lub dowolnie wybrane narzędzie wspomagające tworzenie strony WWW (Pajączek, Zajączek, Notepad++ i wiele innych). Wyniki działania programu obserwować można w dowolnie wybranej przeglądarce. Warto jednak pamiętać, żeby testować dany program w kilku dostępnych przeglądarkach (istnieją różnice w interpretacji).Wymagana jest również podstawowa wiedza z zakresu języka HTML.

Pierwszym sposobem na umieszczenie skryptu w dokumencie HTML jest wprowadzenie go bezpośrednio do kodu przez wykorzystanie znacznika `<script> ... </script>`. Można go wprowadzić w nagłówku **head** lub w głównej części dokumentu **body**.

```
<script type="text/javascript">
kod skryptu
</script>
```

Parametr **type** określa rodzaj języka skryptowego. Nie jest on jednak wymagany. Większość współczesnych przeglądarek zaakceptuje sam znacznik **script**, warto jednak stosować zapis **script** w całości. Natomiast całą treść skryptu (kodu) wprowadza się pomiędzy znaczniki.

Istnieje możliwość umieszczenia skryptu w osobnym pliku. Plik taki nazywamy skryptem zewnętrznym. Może on mieć dowolną nazwę oraz charakterystyczne rozszerzenie **.js** . Plik zewnętrzny zostaje powiązany z dokumentem HTML za pomocą znacznika **script** z dodatkowym oznaczeniem lokalizacji i nazwy pliku wraz z rozszerzeniem.

```
<script type="text/javascript" src="lokalizacja/nazwa.js">
</script>
```

W jednym dokumencie można umieścić kilka skryptów, zarówno osadzonych, jak i zewnętrznych. W przykładzie (list. 1.1) w nagłówku **head** wprowadzono skrypt zewnętrzny, a w głównej części dokumentu **body** – skrypt osadzony.

Listing 1.1

```
<html>
   <head>
   <title>
   Strona ze skryptem JavaScript
   </title>
   <meta http-equiv="Content-Type" content="text/html; charset=
iso-8859-2">
   <meta http-equiv="Content-Language" content="pl">
   <script type="text/javascript" src="lokalizacja/nazwa.js">
   </script>
   </head>
   <body>
<script type="text/javascript">
</script>
   </body>
</html>
```

Obecnie w internecie wiele stron oferuje gotowe przykłady skryptów, które można wykorzystać na stronie, nawet nie znając języka. Mogą to być elementy takie, jak: zegary, kalendarze, kalkulatory, efekty tła, przyciski i wiele innych.

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Przeszukaj dostępne w internecie strony internetowe oferujące gotowe skrypty. Wybierz jeden. Umieść go na stronie i sprawdź jego działanie na różnych przeglądarkach.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Co to jest JavaScript?
2. Jakie znasz sposoby włączenia skryptu do dokumentu HTML?
3. Co jest potrzebne do uruchomienia skryptu napisanego w języku JavaScript?
4. Porównaj działanie kompilatora i interpretera.

2 Instrukcja dokument.write

ZAGADNIENIA

- Za co odpowiada instrukcja dokument.write?
- Jak zmienić wygląd tekstu?
- Jak wyświetlać tekst na stronie za pomocą języka JavaScript?
- Jak stosować elementy języka HTML wewnątrz skryptu?

Kod języka JavaScript zbudowany jest z instrukcji. Instrukcją odpowiadającą za wyświetlenie tekstu na stronie jest **document.write**. Ciąg znaków, który ma zostać wyświetlony, należy umieścić w nawiasie okrągłym oraz w cudzysłowie. Na końcu każdej instrukcji znajduje się średnik:

```
document.write("Wyświetlany tekst");
```

Przed przystąpieniem do opisu instrukcji należy zapoznać się z zapisem ogólnym:

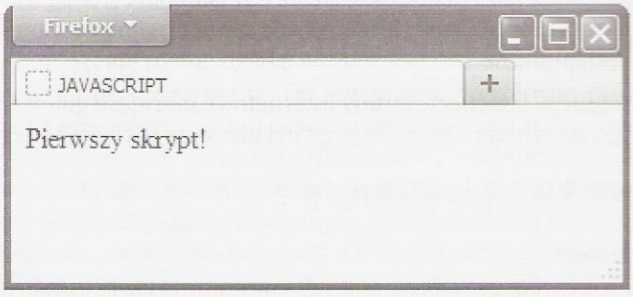
```
obiekt.metoda(argumenty metody);
```

Jak widać, **document** jest obiektem. Reprezentuje on aktualną stronę. Metoda **write** to funkcja działająca na obiekcie **document**, której zadaniem jest wyświetlenie argumentów (tekstu, wartości liczbowych) w oknie przeglądarki.

Pierwszy skrypt (list 2.1) ma za zadanie wyświetlić pojedynczą linijkę tekstu na stronie. Został on umieszczony w głównej części (**body**) dokumentu HTML.

Listing 2.1

```
<script type="text/javascript">
  document.write("Pierwszy skrypt!");
</script>
```



Rys. 2.1. Wynik działania skryptu z list. 2.1

Instrukcja **document.write** pozwala również na wyświetlanie wartości liczbowych. W takim przypadku wystarczy wprowadzić określoną wartość w nawiasie okrągłym bez cudzysłowu:

```
document.write(31);
```

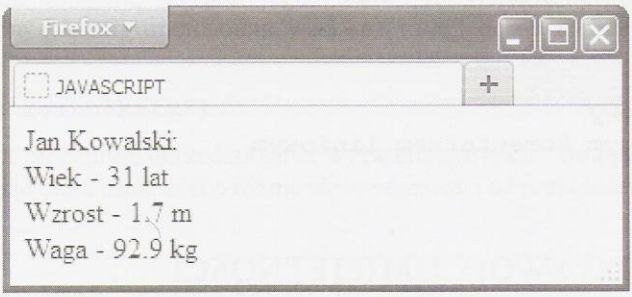
W jednej instrukcji można również umieścić bardziej rozbudowaną treść, składającą się zarówno z tekstu, jak i wartości liczbowych. Należy wówczas zastosować znak + do połączenia obu typów.

```
document.write("Wyświetlany tekst ma "+27+" znaków!");
```

Przykładowy skrypt (list. 2.2) prezentuje wyświetlanie tekstu oraz wartości liczbowych całkowitych i zmiennoprzecinkowych. W przypadku liczb ułamkowych wpisana w skrypcie wartość 1.70 po wyświetleniu na stronie została zaokrąglona do 1.7.

Listing 2.2

```
<script type="text/javascript">
document.write("Jan Kowalski: <br>");
document.write("Wiek - "+31+" lat <br>");
document.write("Wzrost - "+1.70+" m <br>");
document.write("Waga - "+92.9+" kg <br>");
</script>
```



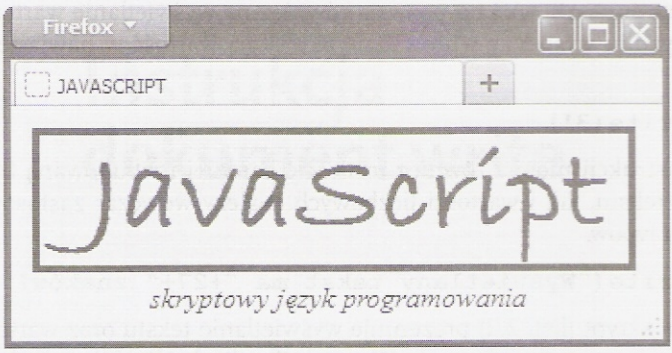
Rys. 2.2. Wynik działa skryptu z list. 2.2

Wprowadzenie kilku instrukcji **document.write** nie spowoduje wyświetlenia ich zawartości w nowej linii. Ponieważ wyświetlana zawartość traktowana jest jak kod HTML, w list. 2.2 wprowadzono dodatkowo znacznik **
**, który odpowiada za przejście do nowej linii.

Wykorzystanie zagnieżdżonych znaczników HTML pozwala w łatwy sposób sformatować informacje wyświetlane za pomocą instrukcji **dokument.write**, a nawet wstawić obrazek (list. 2.3).

Listing 2.3

```
<script type="text/javascript">
document.write("<img src=\"js.png\"<br>");
document.write("<center><i>skryptowy język programowania</center></i>");
</script>
```



Rys. 2.3. Wynik działania skryptu z list. 2.3

Podczas tworzenia skryptów zdarza się, że chcemy ukryć fragment programu lub dodać opis. W tym celu należy wprowadzić **komentarz**. JavaScript daje możliwość zastosowania dwóch rodzajów komentarzy. Pierwszy jest jednowierszowy (liniowy). Rozpoczyna się od znaków `//` i działa do końca danej linii skryptu.

```
// komentarz wierszowy
```

Komentarz wielowierszowy rozpoczyna się od znaków `/*` i kończy się `*/`. W takim komentarzu można umieścić komentarz liniowy, ale zabronione jest zagnieżdżanie w nim komentarzy wielowierszowych.

```
/*
komentarz
wielowierszowy
//z dodatkowym komentarzem liniowym
*/
```

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Zmodyfikuj skrypt z list. 2.2. Wprowadź swoje dane. Zmień formatowanie wyświetlanego tekstu i wprowadź zdjęcie.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Omów instrukcję `dokument.write`.
2. Jak można sformatować tekst wyświetlany przez skrypt?
3. Jak wprowadzić komentarz w skrypcie?
4. W jakim celu stosuje się komentarze w językach programowania?

3 Okno dialogowe

ZAGADNIENIA

- Co to jest okno dialogowe?
- Jakie są rodzaje okien dialogowych?
- Jak wyświetlać okno informacyjne?
- Jak wyświetlać okno decyzyjne?
- Jak wyświetlać okno tekstowe?

Okno dialogowe jest narzędziem pozwalającym na nawiązanie interakcji z użytkownikiem. JavaScript umożliwia wykorzystanie trzech rodzajów okien dialogowych: informacyjnego, decyzyjnego i tekstowego.

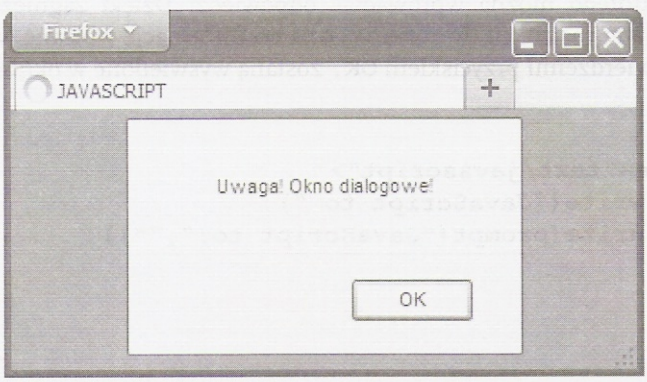
Zadaniem **okna informacyjnego** jest przekazanie określonej informacji. Nie ma ono wpływu na dalsze działanie skryptu. Jego budowa jest wyjątkowo prosta. Wyświetla ono tekst określony jako argument metody `alert` i ma jeden przycisk `OK`, powodujący zamknięcie okna.

```
alert("treść komunikatu");
```

Skrypt z list. 3.1 prezentuje okno dialogowe wyświetlające tekst `"Uwaga! Okno dialogowe!"`. Wygląd okna może nieco różnić się w zależności od przeglądarki internetowej.

Listing 3.1

```
<script type="text/javascript">
alert("Uwaga! Okno dialogowe!");
</script>
```



Rys. 3.1. Wynik działania skryptu z list. 3.1 – okno informacyjne

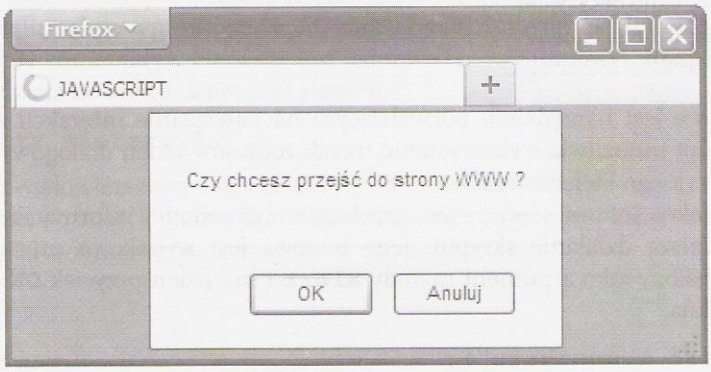
Okno decyzyjne odpowiada za wyświetlenie treści komunikatu stanowiącego argument metody **confirm**. Udostępnia dwa przyciski **OK** oraz **Anuluj**, które po wciśnięciu zwracają wartość logiczną **true** lub **false**.

```
confirm("treść komunikatu");
```

Skrypt z list. 3.2 prezentuje okno decyzyjne wyświetlające tekst **"Czy chcesz przejść do strony WWW?"**. Ponieważ skrypt nie ma żadnej funkcji podpiętej do okna decyzyjnego, wciśnięcie dowolnego klawisza nie wywoła żadnej reakcji.

Listing 3.2

```
<script type="text/javascript">
  confirm("Czy chcesz przejść do strony WWW ?");
</script>
```



Rys. 3.2. Wynik działania skryptu z list. 3.2 – okno decyzyjne

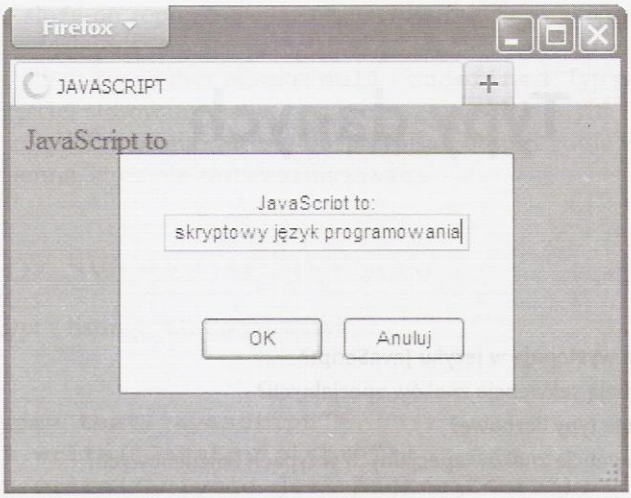
Okno tekstowe wyświetla treść komunikatu stanowiącego argument metody **prompt** oraz pole umożliwiające wprowadzenie danych przez użytkownika. W trakcie wywoływania okna tekstowego w polu może pojawić się tekst domyślny.

```
prompt("treść komunikatu","tekst domyślny");
```

Skrypt list. 3.3 prezentuje okno tekstowe wyświetlające treść **"JavaScript to:"** oraz pole, do którego można wprowadzić odpowiedź. Dzięki zagnieżdżeniu metody **prompt** wewnątrz instrukcji **document.write** informacje wpisane w polu okna tekstowego, po zatwierdzeniu przyciskiem **OK**, zostaną wyświetlone w oknie przeglądarki.

Listing 3.3.

```
<script type="text/javascript">
  document.write("JavaScript to ");
  document.write(prompt("JavaScript to:",""));
</script>
```



Rys. 3.3. Wynik działa skryptu z list. 3.3 – okno tekstowe

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Utwórz skrypt wyświetlający trzy różne okienka dialogowe. Tekst wprowadzony w okienku tekstowym wyświetl w oknie przeglądarki.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Scharakteryzuj różne okna dialogowe.
2. Jak wyświetlić okno dialogowe na stronie?
3. Podaj praktyczne przykłady zastosowania okien dialogowych.

4 Typy danych

ZAGADNIENIA

- **Jakie typy danych występują w języku JavaScript?**
- **Jakie znaczenie mają sekwencje znaków specjalnych?**
- **Jak stosować różne typy liczbowe?**
- **Jak stosować sekwencje znaków specjalnych w typach łańcuchowych?**

Język JavaScript udostępnia kilka typów danych. Typ danych to zbiór wartości, jakie mogą przyjmować dane. Należą do nich: typ liczbowy, łańcuchowy, logiczny, obiektowy oraz typy specjalne.

Typ liczbowy reprezentuje różnego rodzaju liczby. W porównaniu z innymi językami programowania nie uwzględnia on podziału na liczby całkowite i zmiennoprzecinkowe. Umożliwia wprowadzanie liczb w postaci dziesiętnej (np. 12 lub 14), ósemkowej (np. 012) lub szesnastkowej (np. 0xBD). Dozwolona jest również notacja wykładnicza w postaci X.YeZ, gdzie X stanowi część całkowitą, Y – część dziesiętną, a Z jest wykładnikiem potęgi liczby 10 (np. 0.1e2).

Typ łańcuchowy to dowolne ciągi znaków. Należy umieścić je w cudzysłowie lub pomiędzy znakami apostrofów. Mogą dodatkowo zawierać sekwencje znaków specjalnych (tab. 4.1).

Tabela 4.1. Sekwencje znaków specjalnych

Sekwencja	Znaczenie
\b	backspace
\n	nowy wiersz
\r	powrót karetki
\f	nowa strona
\t	tabulacja pozioma
\"	cudzysłów
\'	apostrof
\\	lewy ukośnik

Typ logiczny może przyjmować jedną z dwóch dostępnych wartości: `true` (prawda) oraz `false` (fałsz). Stosowany jest głównie przy budowaniu wyrażeń logicznych lub do porównywania danych.

Typ obiektowy służy do reprezentacji obiektów. Najczęściej wykorzystuje się obiekty wbudowane oraz udostępniane przez przeglądarkę.

Typy specjalne dzielą się na dwa rodzaje: **null** i **undefined**. Typ **null** określa wartość pustą. Najczęściej stosowany jest w programowaniu obiektowym. Typ **undefined** określa wartość niezdefiniowaną. Można go przypisać bezpośrednio do zmiennej lub przyjmuje go zmienna, która nie została zainicjowana.

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Przetestuj skrypt z listingu 4.1. Co zauważyłeś?

Listing 4.1

```
<script type="text/javascript">
   document.write("JavaScript<br>");
   document.write("\"Życie jest piękne\"<br>");
   document.write(15+"<br>");
   document.write(100.4+"<br>");
   document.write(-26+"<br>");
   document.write(0.1e2+"<br>");
   document.write(0xAA+"<br>");
   document.write(-0xCD);
</script>
```

SPRAWDŹ SWOJĄ WIEDZĘ

1. Scharakteryzuj typy danych dostępne w języku JavaScript.
2. Jakie znasz sekwencje znaków specjalnych?

5 Zmienne

ZAGADNIENIA

- Co to jest zmienna?
- Jakie są zasady deklaracji zmiennej w języku JavaScript?
- Jak deklarować zmienne?
- Jak przypisać wartość do zmiennej?

Zmienna jest to element programu pozwalający na przechowywanie danych różnych typów. W języku JavaScript, w odróżnieniu od innych języków programowania, nie wymaga się podania typu zmiennej podczas jej deklaracji. Ponadto typ zmiennej może ulec modyfikacji w trakcie wykonywania skryptu – np. zmiennej typu łańcuchowego możemy przypisać wartość liczbową.

Deklaracja zmiennej odbywa się przez nadanie jej jednoznacznej nazwy, przez którą jest identyfikowana. Przed nazwą należy wprowadzić instrukcję **var**:

```
var nazwa_zmiennej;
```

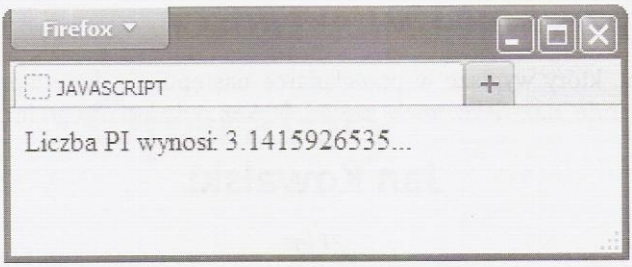
Nazwa zmiennej musi zaczynać się od litery. Wewnątrz jej składni dopuszcza się stosowanie liter, cyfr i znaku podkreślenia (_).

Przykład z list. 5.1 prezentuje deklarację dwóch zmiennych. Zmiennej **zmienna_1** przypisano ciąg znaków umieszczonych w cudzysłowie. Oznacza to, że **zmienna_1** jest typu łańcuchowego. **Zmienna_2** jest typu liczbowego, przypisano jej wartość zmiennoprzecinkową. Za pomocą instrukcji **document.write** wypisano wartości obu zmiennych w oknie przeglądarki (rys. 5.1).

Listing 5.1

```
<script type="text/javascript">
   var zmienna_1="Liczba PI wynosi: ";
   var zmienna_2=3.1415926535;
   document.write(zmienna_1+zmienna_2+"...");
</script>
```

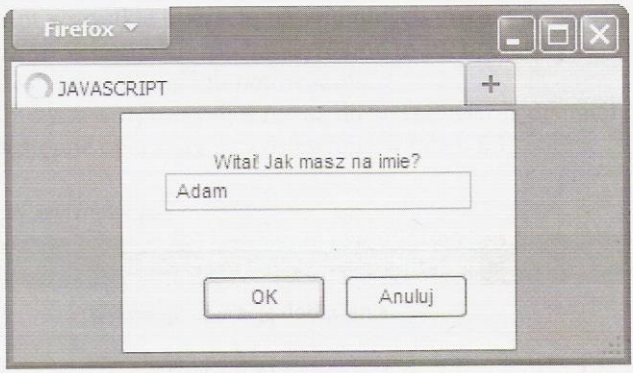
Przykład z list. 5.2 prezentuje, jak przy wykorzystaniu metody **prompt** i okna tekstowego (rys. 5.2) przypisać do zmiennej wartości podane przez użytkownika. Za pomocą instrukcji **document.write** wypisano wartość zmiennej **imie** (wprowadzonej przez użytkownika) w oknie przeglądarki (rys. 5.3). Jeżeli użytkownik nie wprowadzi imienia i wciśnie przycisk **Anuluj**, do zmiennej **imie** zostanie przypisana wartość **null**. Wówczas na ekranie przeglądarki pojawi się napis „Cześć null!".



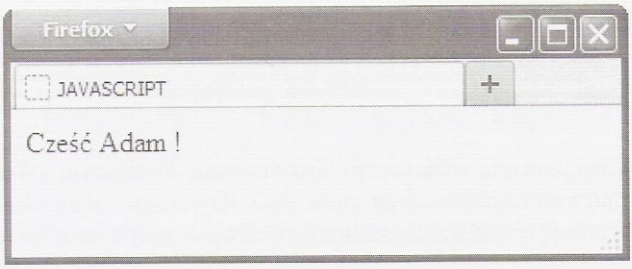
Rys. 5.1. Wynik działaniania skryptu z list. 5.1 – zmienne

Listing 5.2

```
<script type="text/javascript">
  var imie=prompt("Witaj! Jak masz na imię?","");
  document.write("Cześć "+imie+" !");
</script>
```



Rys. 5.2. Wynik działania skryptu z list. 5.2 – okno tekstowe



Rys. 5.3. Wynik działania skryptu z list. 5.2

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Utwórz skrypt, który wypisze w przeglądarce następujące dane wprowadzone przez użytkownika: imię, nazwisko, wiek, zawód. Dokonaj dodatkowego formatowania:

Jan Kowalski

21 lat

`student`

SPRAWDŹ SWOJĄ WIEDZĘ

1. Dlaczego nie określa się typu zmiennej przy jej deklaracji?
2. Czy typ zmiennej może ulec modyfikacji się w trakcie wykonywania skryptu?
3. Jakiej metody użyć w celu przypisania zmiennej wartości podanej przez użytkownika?

6 Operatory

ZAGADNIENIA

- Jakie operatory występują w języku JavaScript?
- Co to jest inkrementacja i dekrementacja?
- Co to jest konkatencja?
- Jak wykonywać operację przypisania?
- Jak stosować operatory arytmetyczne, logiczne, bitowe?
- Jak wykorzystywać w skrypcie inkrementację i dekrementację?

W języku JavaScript wszystkie operacje na zmiennych dokonywane są za pomocą odpowiednich operatorów. Wśród nich można wyróżnić m.in. operatory arytmetyczne, logiczne, bitowe, operatory przypisania lub porównania.

Operatory arytmetyczne wykorzystywane są do wykonywania operacji matematycznych na zmiennych. Dostępne operatory przedstawiono w tabeli 6.1.

Tabela 6.1. Operatory arytmetyczne

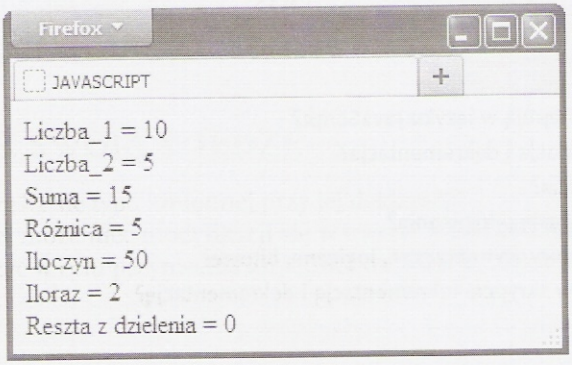
Symbol	Składnia	Opis
+	x+y	Wykonuje operację dodawania.
-	x-y	Wykonuje operację odejmowania.
-	-x	Wykonuje negację zmiennej.
%	x%y	Zwraca resztę z dzielenia pierwszej zmiennej przez drugą (dzielenie modulo).
*	x*y	Wykonuje operację mnożenia.
/	x/y	Wykonuje operację dzielenia.

Przykład z list. 6.1 prezentuje zastosowanie operatorów arytmetycznych na dwóch zadeklarowanych w skrypcie zmiennych. Operatory wprowadzono wewnątrz instrukcji **document.write**, wykorzystując dodatkowy nawias oddzielający operacje matematyczne.

Listing 6.1

```
<script type="text/javascript">
var liczba_1=10;
var liczba_2=5;
document.write("Liczba_1 = "+liczba_1+"<br>");
```

```
document.write("Liczba_2 = "+liczba_2+"<br>");
document.write("Suma = "+(liczba_1+liczba_2)+"<br>");
document.write("Różnica = "+(liczba_1-liczba_2)+"<br>");
document.write("Iloczyn = "+(liczba_1*liczba_2)+"<br>");
document.write("Iloraz = "+(liczba_1/liczba_2)+"<br>");
document.write("Reszta z dzielenia = "+(liczba_1%liczba_2)+"<br>");
</script>
```



Rys. 6.1. Wynik działania skryptu z list. 6.1

Operator łańcuchowy pozwala na złączenie dwóch ciągów znaków w jeden. W języku programowania takie połączenie nazywane jest **konkatencją**.

Tabela 6.2. Operator łańcuchowy (konkatencja)

Symbol	Składnia	Opis
+	"text1"+"text2"	Łączy dwa ciągi znaków w jeden.

Operatory bitowe związane są z wykonywaniem operacji na bitach. Na odpowiednich bitach zmiennych wykonywane są operacje algebry logicznej. Dostępne operatory przedstawiono w tabeli 6.3.

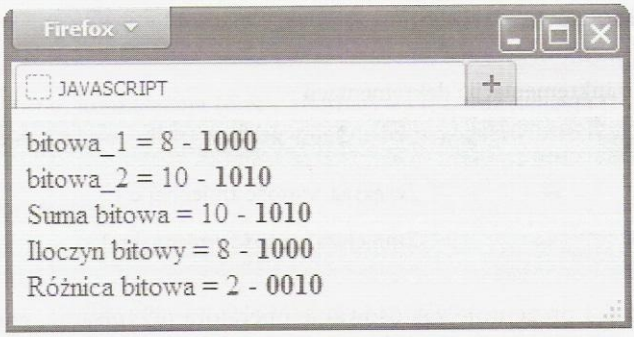
Tabela 6.3. Operatory bitowe

Symbol	Składnia	Opis
&	x&y	Wykonuje bitową operację AND, która wyświetla 1, jeśli obie zmienne wynoszą 1.
^	x^y	Wykonuje bitową operację XOR, która wyświetla 1, jeśli jedna ze zmiennych (ale nie obie jednocześnie) wynosi 1.
|	x|y	Wykonuje bitową operację OR, która wyświetla 1, jeśli jedna ze zmiennych wynosi 1.
<<	x<<y	Wykonuje przesunięcie bitów w lewo o podaną liczbę miejsc.
>>	x>>y	Wykonuje przesunięcie bitów w prawo o podaną liczbę miejsc.

Przykład z list. 6.2 prezentuje działanie sumy bitowej, iloczynu bitowego oraz różnicy bitowej. Zmienne zapisane są w postaci dziesiętnej, ale wykonywane na nich operacje przeprowadzane są na ich odpowiednikach w systemie dwójkowym. Wynik operacji również przedstawiony jest w systemie dziesiętnym. Dla ułatwienia wypisano również wszystkie wartości w systemie dwójkowym wytłuszczoną czcionką.

Listing 6.2

```
<script type="text/javascript">
var bitowa_1=8;
var bitowa_2=10;
document.write("bitowa_1 = "+bitowa_1+" - <b>1000</b><br>");
document.write("bitowa_2 = "+bitowa_2+" - <b>1010</b><br>");
document.write("Suma bitowa = "+(bitowa_1|bitowa_2)+" -
<b>1010</b><br>");
document.write("Iloczyn bitowy = "+(bitowa_1&bitowa_2)+" -
<b>1000</b><br>");
document.write("Różnica bitowa = "+(bitowa_1^bitowa_2)+" -
<b>0010</b><br>");
</script>
```



Rys. 6.2. Wynik działania skryptu z list. 6.2

Podstawowym **operatorem przypisania** jest znak =. Odpowiada on za przypisanie wartości argumentu prawostronnego argumentowi lewostronnemu. Argumentem prawostronnym może być zmienna lub wyrażenie, natomiast argument prawostronny stanowi zmienna, której zadaniem jest przyjęcie nowej wartości. JavaScript oferuje dodatkowo wiele operatorów łączonych zaprezentowanych w tabeli 6.4.

Tabela 6.4. Operatory przypisania

Symbol	Składnia	Opis
=	x=y	Przypisuje wartość y do zmiennej x.
+=	x+=y	Wykonuje przypisanie x=x+y.
-=	x-=y	Wykonuje przypisanie x=x-y.

Symbol	Składnia	Opis
*=	x*=y	Wykonuje przypisanie x=x*y.
/=	x/=y	Wykonuje przypisanie x=x/y.
%=	x%=y	Wykonuje przypisanie x=x%y.
<=	x<=y	Wykonuje przypisanie x=x<y.
^=	x^=y	Wykonuje przypisanie x=x^y.
|=	x|=y	Wykonuje przypisanie x=x|y.
<<=	x<<=y	Wykonuje przypisanie x=x<<y.
>>=	x>>=y	Wykonuje przypisanie x=x>>y.
>>>=	x>>>==y	Wykonuje przypisanie x=x>>>y.

Często stosowanym operatorem jest **inkrementacja** odpowiadająca zwiększeniu danej wartości o jeden. Zapisywana jest za pomocą dwóch plusów „++". Odwrotne działanie daje operator **dekrementacji** zmniejszający wartość danej zmiennej o jeden. Zapisywany jest za pomocą dwóch minusów „--".

Tabela 6.5. Operator inkrementacji i dekrementacji

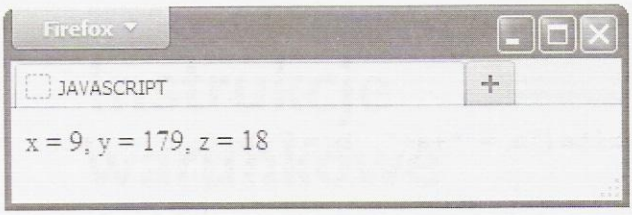
Symbol	Składnia	Opis
++	x++	Zwiększa wartość zmiennej o 1.
--	x--	Zmniejsza wartość zmiennej o 1.

Przykład z list. 6.3 prezentuje zastosowanie operatora przypisania, jednego z operatorów łączonych (*=) oraz operatorów inkrementacji i dekrementacji wobec trzech zadeklarowanych w skrypcie zmiennych.

Listing 6.3

```
<script type="text/javascript">
var x=8;
var y=10;
var z;
z=x+y;
x++;
y*=z;
y--;
document.write("x = "+x+", y = "+y+", z = "+z);
</script>
```

Operatory porównania wykorzystywane są do porównania dwóch argumentów. W wyniku podawana jest wartość `true`, jeżeli zależność jest prawdziwa, lub wartość `false`, jeżeli warunek nie został spełniony.



Rys. 6.3. Wynik działania skryptu z list. 6.3

Tabela 6.6. Operatory porównania

Symbol	Składnia	Opis
!=	x!=y	Zwraca **true**, jeśli zmienne nie są równe.
<	x<y	Zwraca **true**, jeśli pierwsza zmienna jest mniejsza niż druga.
<=	x<=y	Zwraca **true**, jeśli pierwsza zmienna jest mniejsza niż druga lub jej równa.
==	x==y	Zwraca **true**, jeśli zmienne są równe.
>	x>y	Zwraca **true**, jeśli pierwsza zmienna jest większa niż druga.
>=	x>=y	Zwraca **true**, jeśli pierwsza zmienna jest większa niż druga lub jej równa.

Operatory logiczne: konkatencja (&&), alternatywa (||) oraz negacja (!) zwracają wartość **true** (prawda) lub **false** (fałsz) według zależności przedstawionych w tabeli 6.7.

Tabela 6.7. Operatory logiczne

Symbol	Składnia	Opis
!	!x	Ten operator logiczny neguje wyrażenie.
&&	x&&y	Operator logiczny AND zwraca **true**, jeśli obie zmienne są prawdziwe (**true**).
||	x||y	Operator logiczny OR zwraca **true**, jeśli co najmniej jedna ze zmiennych jest prawdziwa (**true**).

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Przeanalizuj skrypt z list. 6.4 i podaj, jakie wartości przyjmują zmienne **a**, **b** i **c** w trakcie działania programu.

Listing 6.4

```
<script type="text/javascript">
var a=7;
var b=10;
var c;
a--;
```

```
c=a/2+5;
b++;
a=c+a/3;
b+=a;
document.write("a = "+a+", b = "+b+", c = "+c);
</script>
```

SPRAWDŹ SWOJĄ WIEDZĘ

1. Wymień i opisz operatory logiczne.
2. Jakie znasz operacje dokonywane na bitach?
3. Co to jest inkrementacja i dekrementacja?

7 Instrukcje warunkowe

ZAGADNIENIA

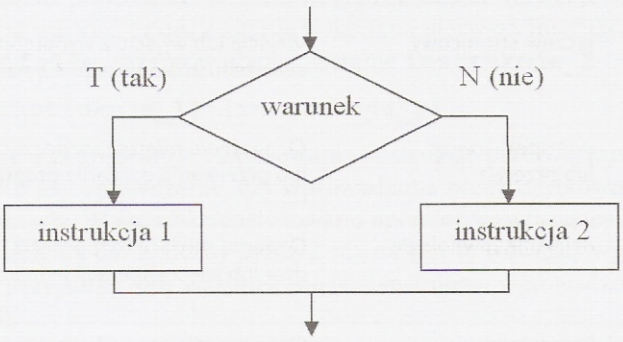
- Jak wygląda ogólny zapis instrukcji warunkowej?
- Co to jest instrukcja przetwarzania warunkowego?
- Jak stosować instrukcję warunkową?
- Jak wykorzystywać operatory porównania w instrukcji warunkowej?

Instrukcja warunkowa określa, który z fragmentów skryptu zostanie wykonany w zależności od spełnienia określonych warunków. Do wyboru mamy uproszczony zapis instrukcji warunkowej, dla której **instrukcje** zostaną wykonane w przypadku, gdy warunek przyjmie wartość **true**:

```
if (warunek) {
instrukcje;
}
```

Bardziej rozbudowana forma instrukcji warunkowej ma dodatkowo element **else**. W tym przypadku **instrukcja_1** zostanie wykonana, gdy warunek przyjmie wartość **true**. W przeciwnym wypadku zostanie wykonana **instrukcja_2**, gdy warunek przyjmie wartość **false**.

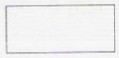
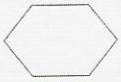
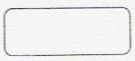
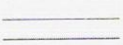
```
if (warunek) {
instrukcja_1;
}
else {
instrukcja_2;
}
```



Rys. 7.1. Schemat blokowy instrukcji warunkowej

Instrukcję warunkową można przedstawić również w postaci schematu blokowego (rys. 7.1). **Schemat blokowy** jest graficzną formą reprezentacji algorytmów. Algorytm stanowi zestaw instrukcji (zadań), których wykonanie prowadzi do osiągnięcia wyznaczonego celu. **Algorytm** przedstawiony w postaci schematu blokowego zbudowany jest z bloków zawierających takie informacje jak: dane wejściowe, wykonywane operacje oraz dane wyjściowe. Do stworzenia schematu blokowego wykorzystuje się odpowiednie bloki opisane w tabeli 7.1.

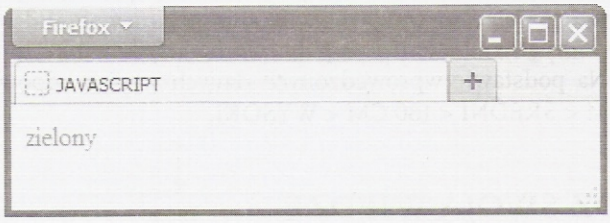
Tabela 7.1. Symbole graficzne w schematach blokowych programów według PN-75/E-01226

Symbol	Nazwa operacji	Wyjaśnienie
	przetwarzanie	Operacja lub grupa operacji, w wyniku których ulega zmianie wartość, postać lub miejsce zapisu danych.
	wprowadzanie lub wyprowadzanie	Wprowadzanie lub wyprowadzanie danych.
	decyzja	Operacja określająca wybór jednej z alternatywnych dróg działania.
	przygotowanie	Modyfikacja rozkazu lub grupy rozkazów powodująca zmianę w przebiegu procesu przetwarzania.
	droga przepływu danych	Więź operacyjna pomiędzy poszczególnymi operacjami procesu przetwarzania.
	skrzyżowanie dróg przepływu danych bez powiązania między nimi	
	łączenie dróg przepływu danych	
	łącznik międzystronicowy	Wejście lub wyjście z wyodrębnionych fragmentów schematu znajdującego się na różnych stronach.
	łącznik stronicowy	Wejście lub wyjście z wyodrębnionych fragmentów schematu znajdującego się na jednej stronie.
	początek, koniec lub przerwa	Oznaczenie miejsce rozpoczęcia, zakończenia lub przerwania działania programu.
	działanie równoległe	Oznacza miejsce rozpoczęcia lub zakończenia dwu lub więcej operacji jednoczesnych.
	komentarz	Oznacza miejsce na komentarz.

Przykład z list. 7.1 prezentuje zastosowanie instrukcji warunkowej. Skrypt w pierwszej kolejności wyświetla okno tekstowe z informacją o wpisaniu koloru. Pierwszy warunek sprawdza, czy wprowadzony napis w oknie tekstowym to **zielony** lub **1**. Jeżeli tak, na ekranie przeglądarki wyświetli się napis **zielony** w kolorze zielonym. W przeciwnym wypadku następuje przejście do sprawdzania drugiego warunku. Jeżeli wprowadzono tekst **czerwony** lub liczbę **2**, na ekranie przeglądarki wyświetli się napis **czerwony** w kolorze czerwonym. W przypadku gdy żaden z warunków nie zostanie spełniony lub użytkownik wybierze przycisk **Anuluj**, na ekranie przeglądarki wyświetli się napis **"Podałeś nieprawidłową wartość!!"**.

Listing 7.1

```
<script type="text/javascript">
var kolor=prompt("Podaj kolor: 1-zielony lub 2-czerwony","");
if((kolor==1|kolor=="zielony")){
document.write("<font color=green>zielony</font>");
}
else if((kolor==2|kolor=="czerwony")){
document.write("<font color=red>czerwony</font>");
}
else{
document.write("Podałeś nieprawidłową wartość!!");
}
</script>
```



Rys. 7.2. Wynik działania skryptu z list. 7.1 po wpisaniu w oknie tekstowym wartości 1

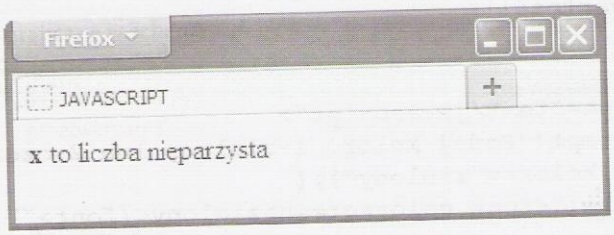
Instrukcja przetwarzania warunkowego pozwala na łatwe i szybkie zastąpienie bloku **if ... else**. Można powiedzieć, że stanowi jego skróconą wersję. Jeżeli wynikiem działania warunku jest wartość **true**, zostanie wykonana pierwsza instrukcja. Jeżeli warunek przyjmie wartość **false**, przeprowadzona zostanie **instrukcja_2**

```
(warunek)?{instrukcja_1}:{instrukcja_2}
```

Przykład z list. 7.2 prezentuje zastosowanie instrukcji przetwarzania warunkowego. Zadaniem skryptu jest sprawdzenie, czy wprowadzona przez użytkownika liczba jest parzysta, czy nieparzysta. W warunku zastosowano operator % zwracający resztę z dzielenia. Jeżeli reszta z dzielenia danej liczby przez 2 równa jest 0, wówczas wprowadzona liczba jest parzysta. W przypadku, gdy warunek nie jest spełniony, wprowadzona liczba jest nieparzysta (rys. 7.3).

Listing 7.2

```
<script type="text/javascript">
var x=prompt("Podaj liczbę","");
var x=(x%2==0)?"parzysta":"nieparzysta";
document.write("<b>x</b> to liczba "+x);
</script>
```



Rys. 7.3. Wynik działania skryptu z list. 7.2 po wpisaniu w oknie tekstowym wartości 15

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Napisz skrypt sprawdzający, czy wprowadzona przez użytkownika wartość jest większa niż 100 i podzielna przez 3.
2. Opracuj skrypt, który wypisze 3 liczby wprowadzone przez użytkownika w kolejności malejącej.
3. Napisz skrypt, którego zadaniem jest pobranie od użytkownika informacji na temat jego wzrostu. Na podstawie wprowadzonych danych wypisz odpowiedni komunikat: NISKI < 150 CM < ŚREDNI < 180 CM < WYSOKI.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Do czego wykorzystywana jest instrukcja warunkowa?
2. Podaj ogólny zapis instrukcji przetwarzania warunkowego.
3. W jakim celu stosuje się schematy blokowe?
4. Omów symbole graficzne wykorzystywane do budowy schematu blokowego

8 Pętle

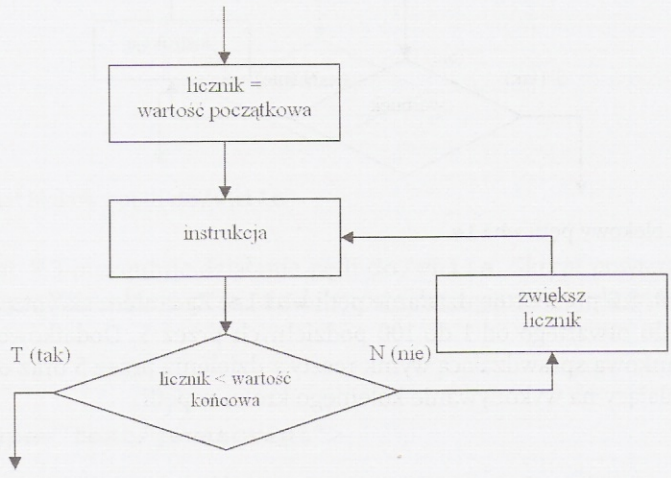
ZAGADNIENIA

- Jakie zadanie mają pętle?
- Jakie pętle są dostępne w języku JavaScript?
- Jak stosować pętle w skrypcie?
- Jak stosować instrukcję warunkową wewnątrz pętli?

Język JavaScript oferuje trzy rodzaje **pętli** (instrukcji iteracyjnych), których zadaniem jest powtarzanie instrukcji określoną liczbę razy. Pętla **for** jest najczęściej stosowanym rodzajem pętli. Składnia pętli jest następująca:

```
for (wyrażenie początkowe; warunek; wyrażenie modyfikujące){
instrukcje;
}
```

Wyrażenie początkowe odpowiada za zainicjowanie zmiennej używanej jako licznik przebiegu pętli. Spełnienie warunku umożliwia wykonanie kolejnego przejścia pętli. Wyrażenie modyfikujące dostosowuje zmienną będącą licznikiem pętli. Pętlę **for** można przedstawić również w postaci schematu blokowego (rys. 8.1).

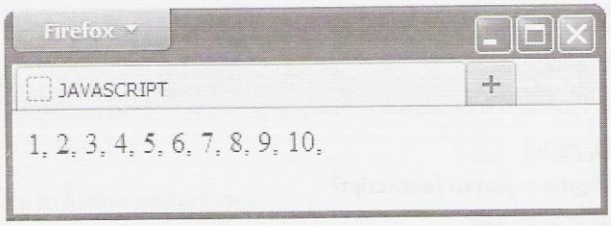


Rys. 8.1. Schemat blokowy pętli **for**

Przykład z list. 8.1 prezentuje wykorzystanie pętli **for**. Za jej pomocą dziesięciokrotnie wykonana została instrukcja **document.write**. Końcowym efektem jest wypisanie w oknie przeglądarki (rys. 8.2) cyfr od 1 do 10.

Listing 8.1

```
<script type="text/javascript">
for (x=1;x<=10;x++)
document.write(x+", ")
</script>
```

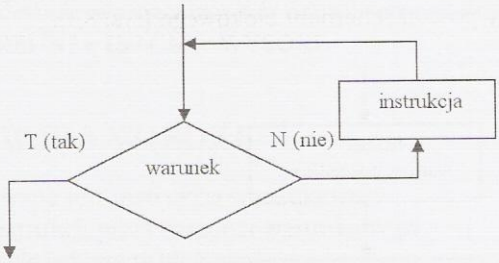


Rys. 8.2. Wynik działania skryptu z list. 8.1 – pętla **for**

Pętla **while** przed wykonaniem instrukcji sprawdza warunek logiczny. Jeżeli warunek przyjmuje wartość **true**, pętla będzie wykonywana do czasu osiągnięcia przez warunek wartości **false**. Może zdarzyć się sytuacja, że pętla nie wykona się ani razu, gdy za pierwszym razem warunek przyjmie wartość **false**. Składnia pętli jest następująca:

```
while (warunek){
instrukcje;
}
```

Pętlę **while** w postaci schematu blokowego prezentuje rys. 8.3.



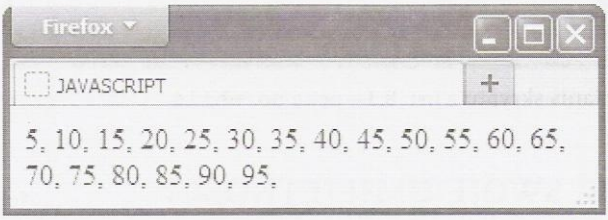
Rys. 8.3. Schemat blokowy pętli **while**

Przykład z list. 8.2 prezentuje działanie pętli **while**. Zadaniem skryptu jest wypisanie liczb z przedziału otwartego od 1 do 100 podzielnych przez 5. Dodatkowo zastosowano instrukcję warunkową sprawdzającą wynik reszty z dzielenia przez 5 oraz operator inkrementacji pozwalający na wykonywanie kolejnego kroku w pętli.

Listing 8.2

```
<script type="text/javascript">
var x=1;
while(x<100)
{
if(x%5==0)
```

```
document.write(x+", ");
x++;
}
</script>
```

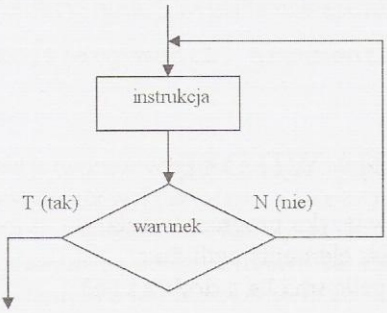


Rys. 8.4. Wynik działania skryptu z list. 8.2 – pętla **while**

Pętla **do/while** pozwala na wykonanie instrukcji przynajmniej raz, zanim zostanie sprawdzony warunek logiczny. Składnia pętli jest następująca:

```
do {
instrukcje;
}
while(warunek)
```

Pętlę **do/while** w postaci schematu blokowego prezentuje rys. 8.5.

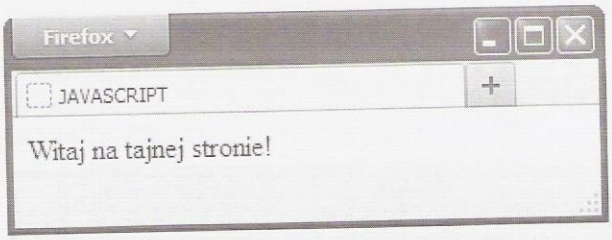


Rys. 8.5. Schemat blokowy pętli **do/while**

Przykład list. 8.3 prezentuje działanie pętli **do/while**. Skrypt powtarza wyświetlanie okna tekstowego. Warunkiem przejścia do strony jest wpisanie w oknie tekstowym hasła **JavaScript**.

Listing 8.3

```
<script type="text/javascript">
do{
var haslo=prompt("Podaj hasło", "")
}
while(haslo!="JavaScript")
document.write("Witaj na tajnej stronie!")
</script>
```



Rys. 8.6. Wynik działania skryptu z list. 8.3 – pętla `do/while`

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Stosując pętle i jedną instrukcję `document.write("#");`, wyświetl w przeglądarce następujące struktury:
   a) linię składająca się z ośmiu #:
   ########
   b) trójkąt o dolnej podstawie pięciu #:
   #
   ##
   ###
   ####
   #####
   c) prostokąt zbudowany ze znaków # o wymiarach 3×7:
   #######
   #######
   #######

SPRAWDŹ SWOJĄ WIEDZĘ

1. Za co odpowiadają pętle w języku programowania?
2. Przedstaw strukturę i opisz elementy pętli `for`.
3. Jaka jest różnica między petlą `while` a `do/while`?

9 Funkcje

ZAGADNIENIA

- Co to jest funkcja?
- Co to są argumenty funkcji?
- Jak definiować funkcję?
- Jak wywoływać funkcję?

Funkcja to zamknięty fragment skryptu oznaczony odpowiednią nazwą, który można wywołać, wielokrotnie odwołując się do tej nazwy. Nazwa funkcji powinna zaczynać się od litery, a kolejne znaki nazwy mogą stanowić litery, cyfry lub znak podkreślenia (_). Funkcja może posiadać argumenty – jeden lub kilka oddzielonych przecinkami. Wywołując funkcję, należy o nich pamiętać. Lista argumentów może być pusta, wówczas pomiędzy nawiasami nic nie zostaje wpisane. W ciele funkcji można wprowadzić dowolną liczbę instrukcji. Należy tylko pamiętać, że instrukcje obowiązkowo należy umieścić w nawiasach klamrowych { }. Ogólnie definicja funkcji wygląda następująco:

```
function nazwa_funkcji(argument1, argument2, ... ,argument n){
instrukcje;
}
```

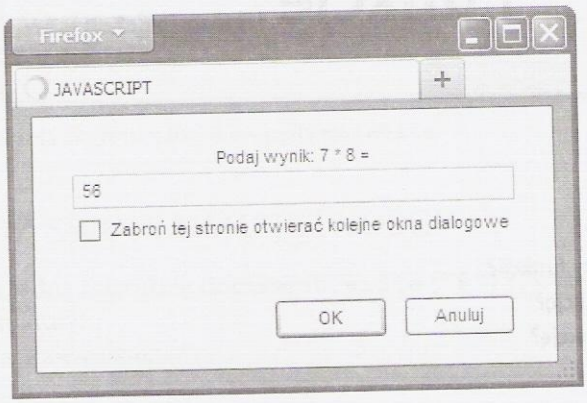
W JavaScript oprócz funkcji tworzonych przez użytkownika istnieją także funkcje wbudowane. Niektóre z nich zostały już wykorzystane: **alert()**, **confirm()**, **prompt()**.

Przykład list. 9.1 prezentuje zastosowanie funkcji w skrypcie. Funkcja ma dwa argumenty, których wartości podawane są przez użytkownika. Zadaniem funkcji jest wyświetlenie okna dialogowego z pytaniem o wynik iloczynu liczb stanowiących argumenty funkcji. Jeżeli wprowadzony wynik jest poprawny, na ekranie przeglądarki pojawi się napis: **Brawo! Wynik poprawny.** W przypadku wprowadzenia błędnej odpowiedzi, wypisana zostanie informacja o błędzie i, dodatkowo, odpowiedź poprawna.

Listing 9.1

```
<script type="text/javascript">
function iloczyn(a,b){
var x=prompt("Podaj wynik: "+a+" * "+b+" ","");
if(x==a*b)
document.write("Brawo! Wynik poprawny.");
else
document.write("Błąd! "+a+" * "+b+" = "+(a*b));
}
x=prompt("Podaj pierwszą liczbę: ");
```

```
y=prompt("Podaj drugą liczbę: ");
iloczyn(x,y);
</script>
```



Rys. 9.1. Wynik działania skryptu z list. 9.1 – funkcja

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Napisz funkcję pobierającą dwa argumenty typu całkowitego x i y (x<y) oraz zwracającą wartość sumy wszystkich elementów przedziału otwartego (x,y).

SPRAWDŹ SWOJĄ WIEDZĘ

1. Podaj przykład funkcji wbudowanej.
2. Jak wywołać funkcję z argumentem? Podaj przykład.
3. W jakim celu stosujemy funkcje?

10 Obiekty

ZAGADNIENIA

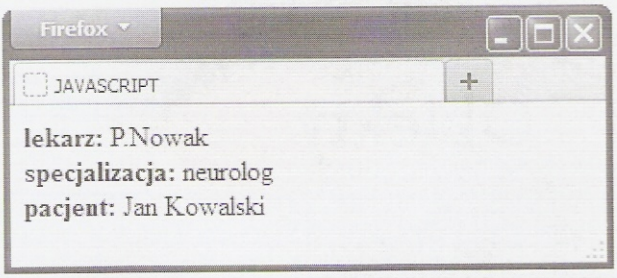
- Co to jest obiekt?
- Za co odpowiada konstruktor?
- Jak tworzyć własne obiekty?
- Jak korzystać z obiektów wbudowanych?

Obiekt to konstrukcja programistyczna mająca swoje cechy charakterystyczne (właściwości), którymi mogą być zmienne lub inne obiekty. Dodatkowo obiekt ma możliwość wykonywania różnych funkcji, które nazywamy metodami. Obiekt definiujemy, tworząc specjalną funkcję zwaną **konstruktorem**, a następnie tworzymy, stosując operator **new**.

Najłatwiej przedstawić to na przykładzie list. 10.1. W skrypcie zdefiniowano za pomocą konstruktora obiekt **osoba** zawierający właściwości: **imie** i **nazwisko** oraz obiekt **szpital** zawierający właściwości: **lekarz**, **specjalizacja** oraz **pacjent** (obiekt). Następnie utworzono oba obiekty za pomocą operatora **new**. Ostatnim krokiem jest wyświetlenie informacji poprzez odwołanie się do konkretnej właściwości obiektu (**nazwa_obiektu.nazwa_właściwości**).

Listing 10.1

```
<script type="text/javascript">
function szpital(lekarz,specjalizacja,pacjent) {
this.lekarz=lekarz;
this.specjalizacja=specjalizacja;
this.pacjent=pacjent;
}
function osoba(imie,nazwisko) {
this.nazwisko=nazwisko;
this.imie=imie;
}
pacjent=new osoba("Jan","Kowalski");
oddzial=new szpital("P.Nowak","neurolog",pacjent);
document.write("<b>lekarz:</b> "+oddzial.lekarz+"<br>");
document.write("<b>specjalizacja:</b> "+oddzial.specjalizacja-
+"<br>");
document.write("<b>pacjent:</b> "+ oddzial.pacjent.imie +"
"+oddzial.pacjent.nazwisko);
</script>
```



Rys. 10.1. Wynik działania skryptu z list. 10.1 – obiekty własne

Oprócz możliwości tworzenia własnych obiektów JavaScrip udostępnia spory zbiór obiektów własnych (wbudowanych), posiadających własne właściwości i metody.

Obiekt window reprezentuje okno przeglądarki i stoi na szczycie hierarchii obiektów. Jest to obiekt domyślny, co oznacza, że do większości jego metod i właściwości można odwołać się bezpośrednio, pomijając jego nazwę. Najpopularniejsze właściwości i metody przedstawiono w tabelach 10.1 oraz 10.2.

Tabela 10.1. Właściwości obiektu window

Właściwości	Opis
`frames[]`	Macierz ramek potomnych w oknie.
`frames.length`	Liczba zdefiniowanych ramek.
`self`	Bieżące okno.
`parent`	Okno rodzicielskie ramki potomnej w zestawie zdefiniowanym znacznikiem `<frameset>`.
`top`	Okno najwyższego rzędu, które jest właścicielem wszystkich widocznych ramek.
`status`	Komunikat pojawiający się w pasku stanu okna przeglądarki.
`defaultStatus`	Komunikat pojawiający się w pasku stanu okna przeglądarki standardowo, kiedy oczekuje ona na wprowadzenie jakichś danych przez użytkownika.
`name`	Wewnętrzny identyfikator okna otwartego metodą `window.open()` (może być niezdefiniowany).

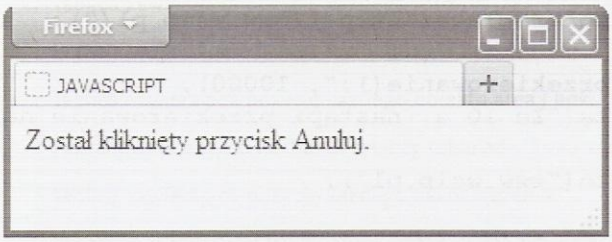
Tabela 10.2. Metody obiektu window

Metody	Opis
`alert("komunikat")`	Wyświetla okno dialogowe.
`confirm("komunikat")`	Wyświetla okno decyzyjne.
`prompt("komunikat")`	Wyświetla okno tekstowe.
`open("URL","nazwa")`	Otwiera na ekranie nowe okno, nadaje mu wewnętrzny identyfikator „`nazwa`" i ściąga do niego dokument wskazany lokalizatorem „`URL`".
`close()`	Zamyka okno z dokumentem.

Przykład z list. 10.2 prezentuje działanie przycisków okna decyzyjnego. Instrukcja warunkowa sprawdza, jaka wartość logiczna została przypisana do zmiennej **test** po wciśnięciu wybranego przycisku w oknie decyzyjnym. Jeżeli zostanie wciśnięty przycisk **OK**, **test** przyjmie wartość **true** i w oknie przeglądarki wyświetli się informacja **Został kliknięty przycisk OK**. Gdy zostanie wciśnięty przycisk **Anuluj**, **test** przyjmie wartość **false** i w oknie przeglądarki pojawi się napis **Został kliknięty przycisk Anuluj** (rys. 10.2).

Listing 10.2

```
<script type="text/javascript">
var test=confirm("Test okienka decyzyjnego:");
if(test==true){
   document.write("Został kliknięty przycisk OK.");
}
else{
   document.write("Został kliknięty przycisk Anuluj.");
}
</script>
```



Rys. 10.2. Wynik działania skryptu z list. 10.2 – wciśnięcie **Anuluj** w oknie decyzyjnym

Obiekt location posiada informacje dotyczące aktualnego adresu URL dokumentu oraz metody pozwalające na operowanie tym adresem. Najpopularniejsze właściwości i metody przedstawiają odpowiednio tabela 10.3 i tabela 10.4.

Tabela 10.3. Właściwości obiektu location

Właściwości	Opis
href	Łańcuch zawierający cały adres URL dokumentu.
protocol	Łańcuch zawierający początek adresu URL wraz z pierwszym dwukropkiem.
host	Łańcuch zawierający nazwę serwera, nazwę domeny.
hostname	Łańcuch zawierający pełną nazwę serwera łącznie z numerem portu.
port	Łańcuch określający używany przez serwer port komunikacyjny.
pathname	Łańcuch zawierający część adresu URL.
hash	Łańcuch rozpoczynający się od znaku #, który określa nazwę zakotwiczenia w dokumencie.
search	Łańcuch rozpoczynający się znakiem ?, który określa zapytanie w adresie URL.

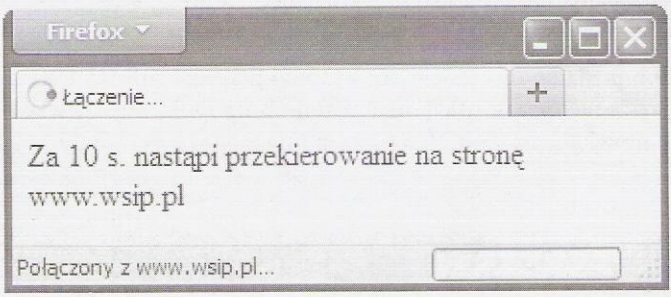
Tabela 10.4. Metody obiektu location

Metody	Opis
`assign(url)`	Wczytuje dokument o adresie wskazanym przez argument url.
`reload(force)`	Wymusza ponowne wczytanie bieżącej strony.
`replace(url)`	Zastępuje bieżący dokument przez wczytany spod adresu wskazanego przez URL.

Przykład z list 10.3 prezentuje metodę **replace** obiektu **location**. Skrypt po upływie 10 sekund powoduje przekierowanie do strony www.wsip.pl. Do opóźnienia czasowego zastosowano funkcję **setTimeout**, która przyjmuje dwa parametry: nazwę funkcji (**przekierowanie()**), która ma zostać wykonana po upływie określonego czasu, oraz czas zwłoki (**10000**) – okres opóźnienia wykonania funkcji podawany w milisekundach.

Listing 10.3

```
<script type="text/javascript">
function przekierowanie(){
window.location.replace("http://www.wsip.pl/");
}
setTimeout("przekierowanie();", 10000);
document.write("Za 10 s. nastąpi przekierowanie na stronę
<br>");
document.write("www.wsip.pl");
</script>
```



Rys. 10.3. Wynik działania skryptu z list. 10.3 – przekierowanie do strony www.wsip.pl

Obiekt document wykorzystuje dostępne metody i właściwości do modyfikacji dokumentu HTML aktualnie wczytanego przez przeglądarkę. Najpopularniejsze właściwości i metody przedstawiono w tabelach 10.5 oraz 10.6.

Przykład z list. 10.4 przedstawia trzy informacje dotyczące skryptu: tytuł dokumentu, datę ostatniej modyfikacji oraz aktualny adres URL. Do wyświetlenia informacji na stronie zastosowano metodę **write**, jedną z najczęściej stosowanych metod obiektu **document**.

Tabela 10.5. Właściwości obiektu **document**

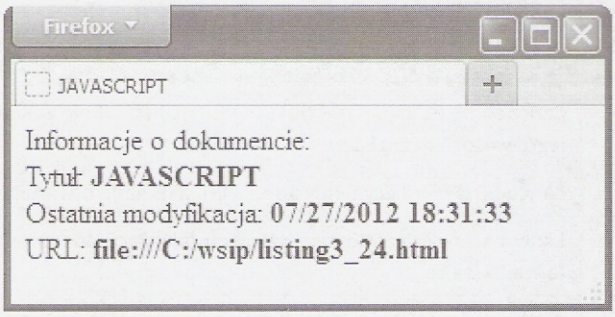
Właściwości	Opis
`title`	Łańcuch określający tytuł dokumentu; jeśli tytuł nie został zdefiniowany, jego wartość to **null**.
`location`	Łańcuch zawierający pełny adres URL aktualnie otwartego dokumentu.
`lastModified`	Łańcuch zawierający datę ostatniej modyfikacji dokumentu; jest on formatu **Date**.
`referrer`	zawiera adres URL, spod którego wywołany został bieżący dokument.
`bgColor`	Łańcuch określający kolor tła dokumentu.
`fgColor`	Łańcuch określający kolor tekstu w dokumencie.
`linkColor`	Łańcuch określający kolor odsyłaczy hipertekstowych w dokumencie.
`vlinkColor`	Łańcuch określający kolor odwiedzonych odsyłaczy hipertekstowych.
`alinkColor`	Łańcuch określający kolor aktywnego odsyłacza hipertekstowego.
`forms[]`	Tablica zawierająca pozycję każdego formularza.
`forms.length`	Przechowuje dane o liczbie formularzy w dokumencie.
`links[]`	Tablica zawierająca pozycję każdego obiektu area i link.
`links.length`	Przechowuje dane o liczbie odsyłaczy (obiektów link i area) w dokumencie.
`anchors[]`	Tablica zawierająca pozycję każdego zakotwiczenia.
`anchors.length`	Przechowuje wartość liczby zakotwiczeń w dokumencie.

Tabela 10.6. Metody obiektu document

Metody	Opis
`write („łańcuch")`	Wypisuje wyrażenie HTML w dokumencie w bieżącym oknie.
`clear()`	Czyści zawartość bieżącego okna.
`close()`	Powoduje zamknięcie bieżącego okna.

Listing 10.4

```
<script type="text/javascript">
document.write("Informacje o dokumencie: <br>");
document.write("Tytuł: <b>"+document.title+"</b><br>");
document.write("Ostatnia modyfikacja: <b>"+document.lastModified+"
</b><br>");
document.write("URL: <b>"+document.location+"</b><br>");
</script>
```



Rys. 10.4. Wynik działania skryptu z list. 10.4 – informacje o dokumencie

Obiekt string stanowi każdy ciąg znaków ujęty w znakach cudzysłowu lub apostrofu. Najpopularniejsze właściwości i metody przedstawiono w tabelach 10.7 i 10.8.

Tabela 10.7. Właściwości obiektu string

Właściwości	Opis
`length`	Zwraca wartość liczbową charakteryzującą liczbę znaków w łańcuchu.

Tabela 10.8. Metody obiektu string

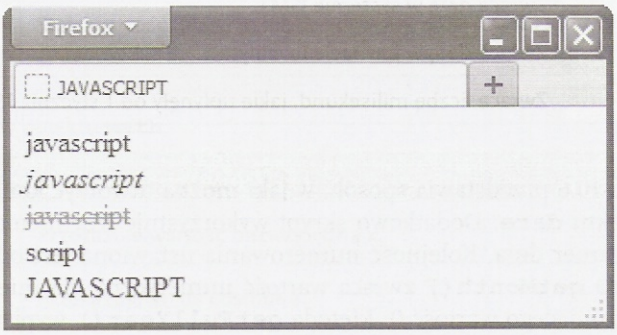
Metody	Opis
`big()`	Zwiększa rozmiar czcionki; odpowiednik znacznika `<big>`.
`blink()`	Tekst migający; odpowiednik znacznika `<blink>`.
`bold()`	Tekst pogrubiony; odpowiednik znacznika `<b>`.
`fixed()`	Odpowiednik znacznika `<tt>`.
`italics()`	Tekst pochylony; odpowiednik znacznika `<i>`.
`small()`	Zmniejsza rozmiar czcionki; odpowiednik znacznika `<small>`.
`sub()`	Odpowiednik znacznika `<sub>`.
`strike()`	Tekst przekreślony; odpowiednik znacznika `<strike>`.
`sup()`	Odpowiednik znacznika `<sup>`.
`fontColor(kolor)`	Ustawia kolor czcionki (tekstu) na wartość `kolor`.
`fontSize(rozmiar)`	Ustawia rozmiar czcionki na wartość `rozmiar`.
`charAt(indeks)`	Zwraca znak z pozycji określonej przez indeks.
`indexOf(podłańcuch [,indeks])`	Przeszukuje łańcuch w poszukiwaniu podłańcucha i zwraca pozycję pierwszego znalezionego znaku.

Metody	Opis
`lastindexOf(podłańcuch [,indeks])`	Przeszukuje łańcuch w poszukiwaniu podłańcucha w kierunku przeciwnym, czyli od końca.
`substring(x, y)`	Zwraca podłańcuch wycięty z łańcucha od pozycji x do pozycji y.
`toLowerCase()`	Konwertuje znaki w łańcuchu na małe litery.
`toUpperCase()`	Konwertuje znaki w łańcuchu na wielkie litery.

Przykład z list. 10.5 przedstawia zastosowanie wybranych metod obiektu **string**. W skrypcie zadeklarowano zmienną **tekst** typu łańcuchowego. Poddano ją formatowaniu z wykorzystaniem jednej lub kilku metod jednocześnie.

Listing 10.5

```
<script type="text/javascript">
var tekst="javascript";
document.write(tekst+"<br>");
document.write(tekst.italics()+"<br>");
document.write(tekst.strike().fontcolor("red")+"<br>");
document.write(tekst.substring(4,10)+"<br>");
document.write(tekst.toUpperCase()+"<br>");
</script>
```



Rys. 10.5. Wynik działania skryptu z list. 10.5 – przykładowe metody obiektu **string**

Obiekt date pozwala na wykonywanie operacji z wykorzystaniem daty i czasu. Pozwala na uzyskanie aktualnej wartości daty i czasu, na korzystanie z ich składowych oraz niezależną zmianę każdej z nich. Praca z obiektem **date** uzależniona jest od użycia konstruktora. Może to być konstruktor bezparametrowy:

```
var data_czas=new Date();
```

lub konstruktor mający od jednego do siedmiu parametrów (rok, miesiąc, dzień, godzina, minuty, sekundy, milisekundy).

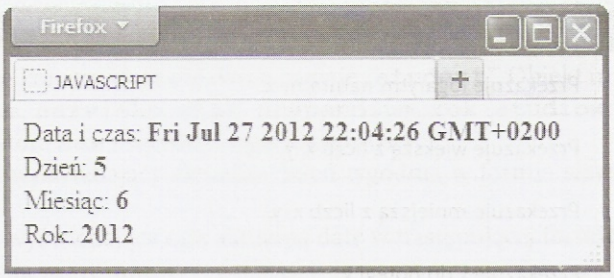
Tabela 10.9. Metody obiektu **date**

Metody	Opis
`getDay()`	Zwraca dzień tygodnia.
`getDate()`	Zwraca dzień miesiąca.
`getHours()`	Zwraca wartość reprezentującą godzinę.
`getMinutes()`	Zwraca wartość reprezentującą minuty.
`getMonth()`	Zwraca wartość reprezentującą numer miesiąca.
`getSeconds()`	Zwraca wartość reprezentującą sekundy.
`getTime()`	Zwraca wartość numeryczną określającą czas; wartość w milisekundach.
`getYear()`	Zwraca rok.
`setDate()`	Ustawia dzień miesiąca.
`setHour()`	Ustawia godzinę.
`setMinutes()`	Ustawia minutę.
`setMonth()`	Ustawia miesiąc.
`setSeconds()`	Ustawia sekundy.
`setTime()`	Ustawia wartość obiektu **Date** – wartość w milisekundach.
`setYear()`	Ustawia rok.
`toGMTString()`	Zwraca datę w systemie GMT.
`toLocaleString()`	Zwraca datę w formacie lokalnym.
`parse(date)`	Zwraca liczbę milisekund, jakie upłynęły od 1 stycznia 1970.

Przykład z list. 10.6 przedstawia sposób, w jaki można utworzyć konstruktor bezparametrowy dla obiektu **date**. Dodatkowo skrypt wykorzystuje trzy metody. Metoda **getDay()** zwraca numer dnia. Kolejność numerowania ustawiona jest od 0 (niedziela) do 6 (sobota). Metoda **getMonth()** zwraca wartość numeryczną dla miesiąca, począwszy od stycznia przyjmującego wartość 0. Metoda **getFullYear()** wypisuje rok w zapisie czterocyfrowym.

Listing 10.6

```
<script type="text/javascript">
var data_czas = new Date();
document.write ("Data i czas: <b>"+data_czas+"</b><br>");
document.write ("Dzień: <b>"+data_czas.getDay()+"</b><br>");
document.write ("Miesiąc: <b>"+data_czas.getMonth()+"</
b><br>");
document.write ("Rok: <b>"+data_czas.getFullYear()+"</b><br>");
</script>
```



Rys. 10.6. Wynik działania skryptu z list. 10.6 – przykładowe metody obiektu `date`

Obiekt `math` wykorzystywany jest do wykonywania różnych obliczeń matematycznych. Udostępnia również szereg stałych matematycznych (tab. 10.10) oraz dodatkowe metody (tab. 10.11).

Tabela 10.10. Właściwości obiektu `math`

Właściwości	Opis
`LN10`	Stała wartość; logarytm naturalny z 10 = 2.302585...
`LN2`	Stała wartość; logarytm naturalny z 2 = 0.693147...
`PI`	Stała wartość; liczba pi = 3.141592...
`SQRT1_2`	Stała wartość; pierwiastek kwadratowy z ½ = 0.707107...
`SQRT2`	Stała wartość; pierwiastek kwadratowy z 2 = 1.414213...

Tabela 10.11. Metody obiektu `math`

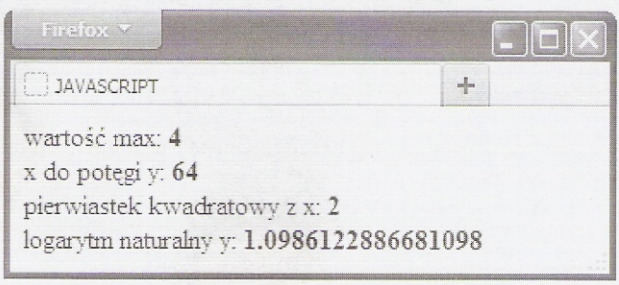
Metody	Opis
`abs(x)`	Przekazuje wartość bezwzględną x.
`acos(x)`	Przekazuje arcus cosinus x.
`asin(x)`	Przekazuje arcus sinus x.
`atan(x)`	Przekazuje arcus tangens x.
`ceil(x)`	Przekazuje najmniejszą liczbę całkowitą większą niż lub równą x.
`cos(x)`	Przekazuje cosinus x.
`exp(x)`	Przekazuje e (stała Eulera) podniesione do potęgi x.
`floor(x)`	Przekazuje największą liczbę całkowitą mniejszą niż lub równą x.

Metody	Opis
`log(x)`	Przekazuje logarytm naturalny x.
`max(x, y)`	Przekazuje większą z liczb x, y.
`min(x, y)`	Przekazuje mniejszą z liczb x, y.
`pow(x, y)`	Przekazuje x do potęgi y.
`round(x)`	Przekazuje x zaokrąglone do najbliższej liczby całkowitej.
`sin(x)`	Przekazuje sinus x.
`sqrt(x)`	Przekazuje pierwiastek kwadratowy z x.
`tan(x)`	Przekazuje tangens x.

Przykład z list. 10.6 przedstawia działanie czterech metod obiektu **`math`**. Skrypt pobiera za pomocą okna tekstowego wartości zmiennych **`x`** i **`y`**. Następnie za pomocą metody **`max(x,y)`** określa, która ze zmiennych ma większą wartość. Kolejna metoda **`pow(x,y)`** podnosi wartość zmiennej **`x`** do potęgi o wartości zmiennej **`y`**. Metoda **`sqrt(x)`** wyciąga pierwiastek kwadratowy ze zmiennej **`x`**. Metoda **`log(y)`** podaje wartość logarytmu naturalnego liczby **`y`**.

Listing 10.7

```
<script type="text/javascript">
var x=prompt("Podaj x:","");
var y=prompt("Podaj y:","");
document.write("wartość max: <b>"+Math.max(x,y)+"</b><BR>");
document.write ("x do potęgi y: <b>"+Math.pow(x,y)+"</b><BR>");
document.write ("pierwiastek kwadratowy z x: <b>"+Math.sqrt(x)+"</b><BR>");
document.write ("logarytm naturalny y: <b>"+Math.log(y)+"</b><BR>");
</script>
```



Rys. 10.7. Wynik działania skryptu z list. 10.7 dla x = 4 i y = 3

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Napisz funkcję konstruktora obiektu o nazwie „**student**". Obiekt ma następujące właściwości: **imie**, **nazwisko**, **wiek**, **stypendium**, **rok_studiow**. Utwórz obiekt za pomocą operatora **new** i wyświetl jego właściwości.
2. Napisz skrypt wyświetlający aktualny dzień tygodnia w formie słownej: poniedziałek, wtorek...
3. Napisz skrypt wyświetlający całą aktualną datę w następującej formie: 21 grudnia 2012.
4. Przygotuj prostą wizytówkę:

   Jan Kowalski
   informatyk
   JKowalski@poczta.pl

   Do sformatowania wyglądu poszczególnych elementów wykorzystaj metody obiektu **string**.
5. Napisz skrypt obliczający wartość bezwzględną liczby wprowadzonej przez użytkownika. W przypadku wprowadzenia błędnych danych wyświetl odpowiedni komunikat (Wprowadzono nieprawidłowe dane!).

SPRAWDŹ SWOJĄ WIEDZĘ

1. Co to jest konstruktor?
2. Jak tworzy się własny obiekt?
3. Jakie znasz obiekty wbudowane?

11 Zdarzenia

ZAGADNIENIA

- **Co to jest zdarzenie?**
- **Jakie rodzaje zdarzeń udostępnia język JavaScript?**
- **Jak wprowadzać wybrane zdarzenie dla dowolnego elementu w skrypcie?**
- **Jak przypisywać funkcje do wybranego zdarzenia?**

Zdarzenia w języku JavaScript to operacje, jakie mogą zachodzić po wystąpieniu określonej sytuacji. Można podzielić je na kilka grup: zdarzenia myszy, zdarzenia klawiatury, zdarzenia dokumentu oraz zdarzenia formularza wraz z jego wszystkimi elementami. Tabela 11.1 zawiera zestaw możliwych zdarzeń związanych w wymienioną grupą.

Tabela 11.1. Lista dostępnych zdarzeń

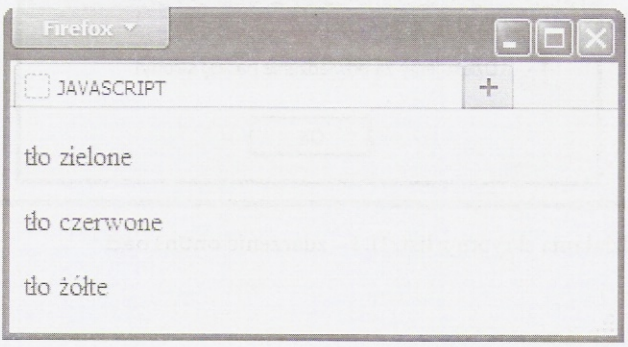
Nazwa	Elementy	Zdarzenie
`onAbort`	obraz	Anulowanie pobrania grafiki.
`onBlur`	okno, elementy formularza	Usunięcie aktywności pola.
`onChange`	text, textarea, select	Zmiana wartości.
`onClick`	dowolny znacznik	Kliknięcie dowolnego znacznika.
`onDblClick`	dowolny znacznik	Podwójne kliknięcie na dowolny znacznik.
`onDragDrop`	okno	Przeciągnięcie obiektu poza okno.
`onError`	obraz, okno	Błąd podczas ładowania.
`onFocus`	okno, elementy formularza	Uaktywnienie elementu.
`onKeyDown`	dokument, obraz, link, textarea	Wciśnięcie (przytrzymanie) klawisza na klawiaturze.
`onKeyPress`	dokument, obraz, link, textarea	Wciśnięcie (krótkotrwałe) klawisza na klawiaturze.
`onKeyUp`	dokument, obraz, link, textarea	Puszczenie klawisza.
`onLoad`	ciało dokumentu	Ładowanie strony.
`onMouseDown`	dokument, przycisk, link	Wciśnięcie (przytrzymanie) klawisza myszy.
`onMouseOut`	wszystko	Opuszczenie danego obszaru przez kursor myszy.

Nazwa	Elementy	Zdarzenie
onMouseOver	wszystko	Najechanie kursorem myszy na dany obszar
onMouseUp	wszystko	Puszczenie klawisza myszy
onMove	okno	Poruszenie oknem
onReset	formularz	Wyczyszczenie formularza
onResize	okno	Zmiana rozmiaru okna
onSelect	text, textarea	Wybranie danego elementu
onSubmit	formularz	Zatwierdzenie formularza
onUnload	ciało dokumentu	Zamknięcie strony

Przykład z list. 11.1 przedstawia zastosowanie zdarzenia **onMouseOver**. W głównej części dokumentu wprowadzono trzy akapity **p**. Po najechaniu wskaźnikiem myszki na wybrany akapit zostanie wywołane zdarzenie powiązane z funkcją **kolor_tla(kolor)**. Funkcja zawiera jeden parametr określający kolor, jaki ma być ustawiony dla zmiany tła dokumentu.

Listing 11.1

```
<body>
<script type="text/javascript">
function kolor_tla(kolor){
document.bgColor=kolor;
}
</script>
<p onMouseOver="kolor_tla("green")">tło zielone</p>
<p onMouseOver="kolor_tla("red")">tło czerwone</p>
<p onMouseOver="kolor_tla("yellow")">tło żółte</p>
</body>
```

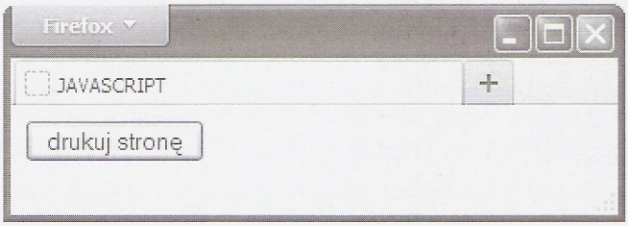


Rys. 11.1. Wynik działania skryptu z list. 11.1 – zdarzenie **onMouseOver**

Przykład z list. 11.2 wykorzystuje zdarzenie **onClick**. W głównej części dokumentu utworzono przycisk (rys. 11.2) za pomocą znacznika **button**. Jeżeli użytkownik kliknie w przycisk, wywoła się funkcja **drukuj()** przypisana do zdarzenia **onClick**. Zadaniem funkcji jest wywołanie okna **drukowanie** pozwalającego na wydruk zawartości strony.

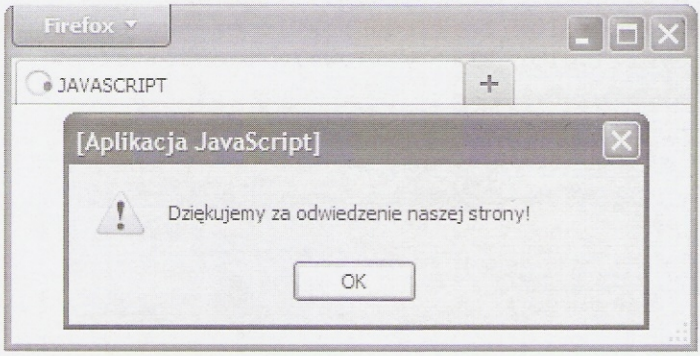
Listing 11.2

```
<body>
<script type="text/javascript">
function drukuj(){
window.print();
}
</script>
<button onClick="drukuj()">drukuj stronę</button>
</body>
```



Rys. 11.2. Wynik działania skryptu z list. 11.2 – zdarzenie **onClick**

Przykład z list. 11.3 przedstawia zastosowanie zdarzenia **onUnload**. Zdarzenie wprowadzono wewnątrz znacznika **body**. Przypisano mu funkcję **zegnaj()**, która podczas opuszczania strony wyświetli okno dialogowe z napisem **Dziękujemy za odwiedzenie naszej strony!**.



Rys. 11.3. Wynik działania skryptu z list. 11.3 – zdarzenie **onUnload**

Listing 11.3

```
<body onUnload="zegnaj()">
<script type="text/javascript">
function zegnaj(){
```

```
alert("Dziękujemy za odwiedzenie naszej strony!");
}
</script>
</body>
```

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Napisz skrypt, który wykorzystując zdarzenia, spowoduje zwiększenie rozmiaru napisu **`Witaj na stronie`** po najechaniu na niego myszką. Po opuszczeniu obszaru tekst wraca do poprzedniego rozmiaru.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Jakie wyróżnia się grupy zdarzeń?
2. Podaj kilka przykładów zdarzeń z użyciem myszki.
3. Podaj kilka przykładów zdarzeń z użyciem klawiatury.

12 Obsługa formularzy

ZAGADNIENIA

- Jakie zdarzenia stosować na formularzach?
- Jak zabezpieczyć formularz?
- Jak stosować zdarzenia związane z formularzem?
- Jak sprawdzać poprawność formularza?

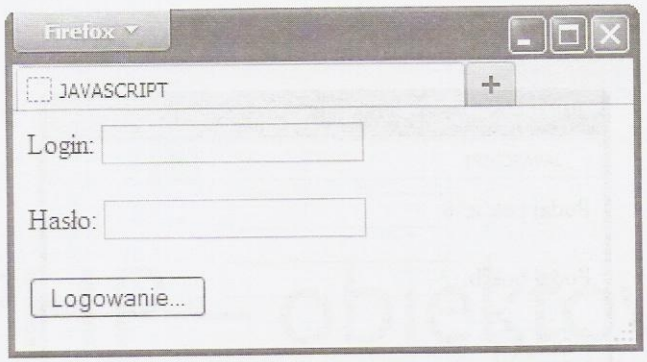
Formularze na stronie internetowej wykorzystywane są głównie do zbierania informacji od użytkownika i przesyłania ich na serwer. JavaScript pozwala na wykorzystanie danych wprowadzanych do formularza po stronie klienta. Umożliwia ich odczytanie, modyfikację i przeprowadzanie na danych dodatkowych operacji, np. matematycznych. Pomaga w sprawdzeniu, czy wszystkie elementy formularza zostały uzupełnione. Zdarzenia związane z obsługą formularzy przedstawiono w poprzednim rozdziale w tabeli 11.1.

Przykład z list. 12.1 przedstawia formularz logowania posiadający pole tekstowe, pole typu hasło oraz przycisk. Oba pola formularza mają nadaną odpowiednią nazwę, przez którą można odwołać się do wprowadzonych danych. Do przycisku przypisano zdarzenia `onClick`, które po jego wciśnięciu powoduje wywołanie funkcji `logowanie(this.form)`. Funkcja jako parametr przyjmuje obiekt reprezentujący bieżący formularz. Jej zadaniem jest sprawdzenie, czy do formularza wprowadzono odpowiednie dane (login – kowalski oraz hasło – zaq123), które pozwalają na przejście do strony `strona_glowna.html`. Jeżeli dane zostaną wprowadzone błędnie lub nie zostaną wprowadzone wcale, strona zawierająca formularz zostanie załadowana ponownie.

Listing 12.1

```
<body>
<script type="text/javascript">
function logowanie(formularz){
if (formularz.login.value=="kowalski"&formularz.pass.value=
="zaq123")
window.location.replace("strona_glowna.html");
else
window.location.replace("");
}
</script>
<form name="formularz1">
Login: <input type="text" name="login"><br><br>
Hasło: <input type="password" name="pass"><br><br>
```

```
<input type="button" value="Logowanie..." onClick="logowanie-
(this.form)">
</form>
</body>
```



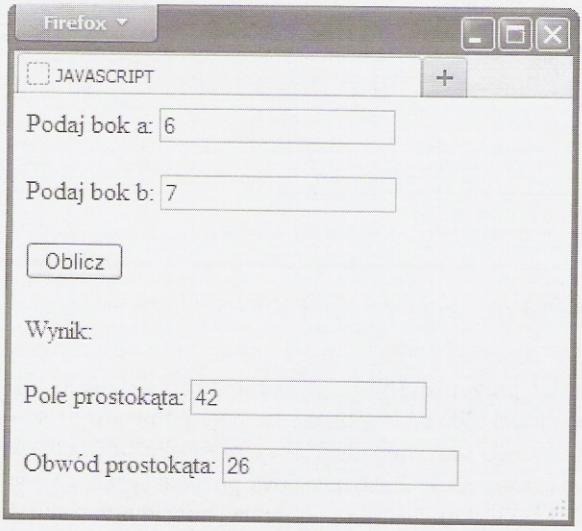
Rys. 12.1. Wynik działania skryptu z list. 12.1 – formularz logowania

Przykład z list. 12.2 prezentuje wykorzystanie formularza do wykonywania operacji matematycznych po stronie klienta – obliczenie pola i obwodu prostokąta. Formularz zawiera cztery pola tekstowe oraz przycisk. Dwa pierwsze pola odpowiadają za pobranie długości boków prostokąta: a i b. Pozostałe dwa pozwalają na wyświetlenie w nich wyników obliczeń. Obliczenia wykonywane są przez wciśnięcie przycisku, który powiązany jest ze zdarzeniem **onClick**. Zdarzenie wywołuje funkcję **licz(this.form)**. Funkcja w pierwszej kolejności sprawdza, czy pola formularza zostały uzupełnione. Jeśli nie, na ekranie pojawi się okno dialogowe z informacją **Wprowadź dane**. Pobrane dane zostają przypisane odpowiednio do zmiennych **a** i **b**, a następnie obliczone na ich podstawie pole i obwód przypisywane są do pól formularza o nazwie **p** i **o**.

Listing 12.2

```
<body>
<script type="text/javascript">
function licz(formularz){
if (formularz.a.value==""|formularz.b.value=="")
alert("Wprowadź dane");
else{
a=formularz.a.value;
b=formularz.b.value;
formularz.p.value=a*b;
formularz.o.value=2*a+2*b;
}
}
</script>
<form name="formularz1" action="">
Podaj bok a: <input type="text" name="a"><br><br>
Podaj bok b: <input type="text" name="b"><br><br>
```

```
<input type="button" value="Oblicz" onClick="licz(this.form)
"><br><br>
Wynik:<br><br>
Pole prostokąta: <input type="text" name="p"><br><br>
Obwód prostokąta: <input type="text" name="o">
</form>
</body>
```



Rys. 12.2. Wynik działania skryptu z list. 12.2 – pole i obwód prostokąta

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Napisz skrypt zawierający formularz pozwalający na obliczenie pola i objętości różnych brył geometrycznych (kula, stożek, sześcian).

SPRAWDŹ SWOJĄ WIEDZĘ

1. Wymień zdarzenia związane z formularzem.
2. Jak sprawdzić, czy użytkownik uzupełnił pola formularz?
3. Na czym polega wykorzystanie języka JavaScript dla poprawnego korzystania z formularza. Podaj praktyczny przykład.

II. PHP – obiektowy, skryptowy język programowania

- Ogólne cechy języka
- Instalacja Apache, PHP i bazy danych MySQL
- Konfiguracja PHP
- Umieszczanie kodu PHP w dokumencie
- Instrukcje: print (), echo ()
- Typy danych
- Zmienne i stałe
- Instrukcje warunkowe
- Pętle
- Funkcje
- Instrukcje dołączania plików
- Tablice
- Programowanie obiektowe
- Obsługa wyjątków
- Obsługa plików
- Obsługa formularzy
- Współpraca PHP i MySQL
- Zabezpieczanie witryn WWW

13 Ogólne cechy języka

ZAGADNIENIA

- **Co to jest PHP?**
- **Jakie możliwości daje język PHP?**
- **Co można stworzyć za pomocą języka PHP?**

PHP jest obiektowym językiem programowania wykonywanym po stronie serwera. Skrypty zagnieżdżane są zwykle w dokumencie HTML lub XHTML, więc do jego stworzenia wystarczy prosty edytor tekstu. Za działanie skryptu odpowiada serwer, który w pierwszej kolejności interpretuje skrypt PHP, a następnie generowana jest strona przesyłana do przeglądarki.

Początki języka PHP sięgają 1994 roku, kiedy to został rozpowszechniony pod nazwą PHP/FI (*Personal Home Page/Forms Interpreter*). Jego twórcą jest Rasmus Lerdorf. W 1995 roku udostępnił on publicznie kod źródłowy PHP Tool 1.0. W 1998 roku dwaj izraelscy programiści, Zeev Supraski i Andi Gutmas, rozpoczęli pracę nad rozszerzeniem możliwości języka i utworzyli wersję PHP3 mającą całkowicie nową architekturę oraz elementy programowania obiektowego. Na tym nie zakończyły się ich działania. Dalsza praca zaowocowała powstaniem wersji PHP4 i obecnie najczęściej stosowanej PHP5, która jest nadal udoskonalana.

Język PHP powstał na podstawie języków programowania C / C++, Pearl oraz Java, lecz jego budowa jest znacznie uproszczona. Jego zadaniem jest rozszerzenie funkcji strony internetowej. Pozwala na tworzenie zarówno prostych, jak i rozbudowanych serwisów. Oferuje programiście wiele dodatkowych możliwości, takich jak:

- współpraca z wieloma popularnymi bazami danych (MySQL),
- obsługa protokołów sieciowych (SSL, IMAP, SMTP),
- wsparcie standardu XML,
- funkcje kryptograficzne,
- zaawansowany moduł programowania obiektowego,
- algorytmy kompresji (GZip),
- narzędzia obsługujące model DOM (obiektowy model dokumentu),
- obsługa przestrzeni nazw i wiele innych.

Dzięki tak wielu dodatkowym możliwościom, a zwłaszcza pracy po stronie serwera, język PHP nadaje się do tworzenia projektów potrafiących zarządzać dużą ilością danych. Mogą to być systemy zarządzania treścią, fora dyskusyjne, sklepy internetowe, komunikatory, galerie, hostingi i wiele innych.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Na jakich językach programowania opiera się PHP?
2. Jakie możliwości udostępnia PHP?

14 Instalacja Apache, PHP i bazy danych MySQL

ZAGADNIENIA

- Co to jest interpreter PHP?
- Jakie są dostępne serwery lokalne?
- Jak zainstalować serwer lokalny?
- Jak korzystać z serwera lokalnego?

Do tworzenia stron wzbogaconych o elementy zbudowane na podstawie języka PHP wystarczy dowolny edytor tekstu (np. Notatnik) lub inny program wspomagający tworzenie stron (Zajączek, Pajączek). Aby sprawdzić działanie takiej strony, należy umieścić ją na serwerze wyposażonym w obsługę PHP. Przy tworzeniu rozbudowanych stron może to być nieco kłopotliwe. Dlatego warto skorzystać z lokalnego serwera WWW, który można zainstalować na dowolnym komputerze.

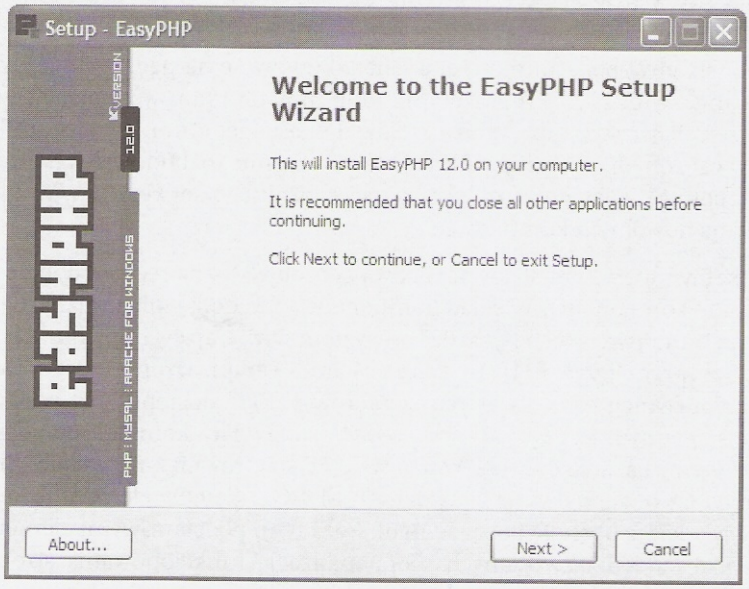
Serwer lokalny można zainstalować na dwa sposoby. Można ręcznie dobrać i skonfigurować wszystkie komponenty. W skład komponentów wchodzi kilka ważnych elementów. Pierwszy z nich to **interpreter języka PHP** – system zawierający zbiór funkcji i modułów udostępnianych przez język PHP (w zależności od wersji). Drugim jest **serwer Apache**, jeden z najpopularniejszych otwartych serwerów HTTP, dostępny dla wielu systemów operacyjnych. Kolejnym jest **baza danych MySQL** stanowiąca kompleksowy system zarządzania relacyjnymi bazami danych. Warto wykorzystać również narzędzie **phpMyAdmin** pozwalające na łatwe zarządzanie bazą danych MySQL. Kolejny element to **serwer SMTP** umożliwiający lokalną obsługę poczty e-mail. Poza tymi podstawowymi elementami można dobrać kilka narzędzi związanych z optymalizacją funkcjonowania strony, kontrolą logów serwera czy obsługą FTP.

Łatwiejszym sposobem jest skorzystanie z narzędzi zawierających już w sobie wszystkie potrzebne elementy. Są to gotowe serwery lokalne, odpowiednio skonfigurowane, które w prosty sposób można zainstalować na komputerach lokalnych. Obecnie dostępnych jest wiele serwerów różniących się doborem wersji komponentów i konfiguracją. Tabela 14.1 przedstawia kilka propozycji serwerów oraz adresy stron internetowych, z których można je pobrać.

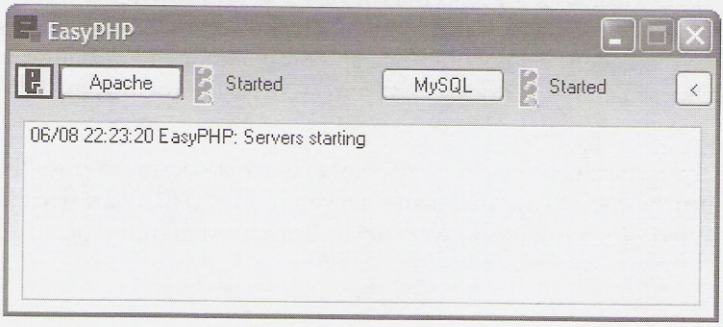
Instalacja serwera lokalnego polega na pobraniu pliku instalacyjnego (EasyPHP-setup.exe), który należy uruchomić. Pojawia się panel powitalny (rys. 14.1), określający instalowaną wersję oraz sugerujący zamknięcie wszystkich innych programów przed rozpoczęciem instalacji.

Tabela 14.1. Lokalne serwery WWW

Nazwa	Adres
Appserv	http://www.appservnetwork.com
EasyPHP	http://www.easyphp.org/
FoxServ	http://sourceforge.net/projects/foxserv/
Krasnal Serv	http://www.krasnal.tk/
XAMPP	http://www.apachefriends.org/en/xampp.html



Rys. 14.1. Instalacja lokalnego serwera EasyPHP



Rys. 14.2. Główne okno programu EasyPHP

Należy zaakceptować zapoznanie się z licencjami programów składowych serwera (m.in. PHP, Apache, MySQL, phpMyAdmin, GNU), a następnie określić folder, w którym zostanie zainstalowany serwer, lub zatwierdzić ścieżkę domyślną (C:\Program files\EasyPHP-12.0). Po zatwierdzeniu następuje instalacja, po której zakończeniu dochodzi do automatycznego uruchomienia serwera. Jeżeli wszystko przebiegło pomyślnie, w zasobniku systemowym (pasek koło zegarka) pojawi się ikonka w postaci czarnej literki e. Kliknięcie jej spowoduje wyświetlenie okienka programu (rys. 14.2). Poprawność działania elementów programu określają symbole „sygnalizacji świetlnej". Oba światełka świecące na zielono oznaczają, że wszystko jest dobrze i można zacząć programowanie.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Jakie znasz darmowe serwery lokalne?
2. Co to jest Apache?
3. Za co odpowiada interpreter języka PHP?

15 Konfiguracja PHP

ZAGADNIENIA

- **Co można konfigurować w lokalnym serwerze WWW?**
- **Jakie pliki podlegają konfiguracji?**
- **Jak konfigurować serwer lokalny?**

Zainstalowany lokalny serwer WWW po części jest już skonfigurowany. Wstępne ustawienia pozwalają na rozpoczęcie pracy. Warto jednak zapoznać się z niektórymi funkcjami, jakie daje serwer.

Pierwszym plikiem, który można skonfigurować, jest **php.ini**. Jest to plik pozwalający zdefiniować ustawienia PHP dla całego serwera. Plik zbudowany jest z dyrektyw, którym przypisano określone wartości. Jeżeli w pliku chcemy umieścić komentarz lub ukryć dyrektywę, wstawiamy znak średnika (;) dla każdej linii.

Zmianą, którą warto wprowadzić w pliku **php.ini**, jest ustawienie opcji **error_reporting** tak, aby serwer reagował na wszelkie występujące w skrypcie błędy:

```
error_reporting = E_ALL | E_STRICT
```

Kolejnym krokiem jest upewnienie się, czy opcja **display_errors**, zezwalająca na wyświetlanie błędów, dla danej wersji PHP jest włączona:

```
display_errors = ON
```

W przypadku wykorzystywania skróconej wersji tagów **<? ?>** należy włączyć opcję: **short_open_tag = ON**

Opis innych przykładowych dyrektyw pozwalających na wprowadzenie zmian w konfiguracji pliku **php.ini** prezentuje tabela 15.1.

Tabela 15.1. Dyrektywy konfiguracji pliku **php.ini**

Dyrektywa	Wartość domyślna	Opis
allow_url_fopen	On	Możliwość stosowania funkcji: file_get_contents fopen
asp_tags	Off	Możliwość użycia tagów <% %> zamiast <?php ?>.
display_errors	On	Wyświetlanie błędów.
error_log	no value	Lokalizacja zapisu („/logs/php_errors.log").

Dyrektywa	Wartość domyślna	Opis
error_reporting	Off	Rodzaj logowanych błędów (E_ALL, E_NOTICE, E_STRICT).
log_errors	Off	Logowanie błędów.
memory_limit	128m	Limit pamięci dla wykonywanych skryptów.
post_max_size	64M	Wielkość danych przesyłanych metodą POST.
safe_mode	Off	Wprowadzenie safe_mode.
short_open_tag	On	Zezwala na stosowanie krótszego tagu <? ?>.
upload_max_filesize	64M	Wartość mniejsza lub równa wartości post_max_size.
upload_tmp_dir	no value	Zdefiniowanie tymczasowego katalogu do wysyłania plików (innego niż /tmp).
session.cookie_domain	no value	Domena obsługiwania sesji.
session.save_path	/tmp	Katalog przechowujący pliki sesji.

Plik **`httpd.conf`** zawiera domyślną konfigurację serwera. Plik zbudowany jest z dyrektyw, którym przypisano określone wartości. Jeżeli w pliku chcemy umieścić komentarz lub ukryć dyrektywę, wstawiamy znak **#** dla każdej linii. W przypadku wystąpienia błędów przy uruchamianiu serwera należy sprawdzić, czy inny z zainstalowanych na komputerze programów nie nasłuchuje na porcie 8080 lub zmienić wpis w konfiguracji pliku **`httpd.conf`**:

```
Listen 8080
```

Dyrektywa ta nakazuje serwerowi nasłuchiwanie dla więcej niż jednego adresu lub portu, może być stosowana więcej niż tylko raz.

Opis przykładowych dyrektyw prezentuje tabela 15.2.

Tabela 15.2. Dyrektywy konfiguracji pliku httpd.conf

Dyrektywa	Opis
AccessFileName	Określa nazwę pliku konfiguracyjnego odczytywanego przy każdym żądaniu.
AddDefaultCharset	Kodowanie dokumentów.
AddLanguage	Mapuje rozszerzenia plików na konkretny język.
BrowserMatch	Ustawia odpowiednie zmienne środowiskowe.
ErrorDocument	Określa dokumenty, które zostaną przesłane do użytkownika w przypadku wystąpienia błędu.
Include	Dołącza do konfiguracji określone pliki.
LanguagePriority	Hierarchia ważności poszczególnych języków.
Listen	Określa nasłuchiwanie na wybranych portach (np. 80, 8080).
LogFormat	Określa formaty logowania.
LogLevel	Określa, jakie zdarzenia będą logowane w dzienniku.
NameVirtualHost	Określa adres IP serwera wirtualnego (127.0.0.1).

Plik **my.ini** odpowiada za konfigurację bazy danych MySQL. Plik zbudowany jest z dyrektyw, którym przypisano określone wartości. Jeżeli w pliku chcemy umieścić komentarz lub ukryć dyrektywę, wstawiamy znak # dla każdej linii.

Konfigurację bazy danych warto dostosować do parametrów fizycznych serwera, zwłaszcza uwzględniając pamięć fizyczną. Przykładowa optymalizacja podstawowych parametrów bazy dla małej pamięci fizycznej, a zwiększonej liczby zapytań do bazy danych przedstawia się następująco:

```
query_cache_size=36M
table_cache=512
```

Opis przykładowych dyrektyw prezentuje tabela 15.3.

Tabela 15.3. Dyrektywy konfiguracji pliku **my.ini**

Dyrektywa	Opis
default-character-set	Określa sposób kodowania znaków(utf-8, latin2).
max_connections	Określa maksymalną liczbę połączeń w jednym czasie.
port	Określa port nasłuchu (3306).
query_cache_size	Ustawia rozmiar kwerendy w pamięci podręcznej.
table_cache	Liczba otwartych tabel dla wszystkich wątków.
thread_cache_size	Liczba wątków przechowywanych w pamięci podręcznej.
tmp_table_size	Maksymalny rozmiar wewnętrznych tabel tymczasowych.

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Zmień konfigurację pliku **php.ini**, tak aby możliwe było stosowanie krótszego tagu **<? ?>**.
2. Zmień konfigurację pliku **httpd.conf**, tak aby serwer nasłuchiwał na porcie 8080.
3. Zmień konfigurację pliku **my.ini**, tak aby kodowanie znaków ustawione było na latin2.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Jakie pliki można poddać konfiguracji?
2. Jak i gdzie wyłączyć wyświetlanie błędów?
3. Co to są roboty internetowe?

16 Umieszczanie kodu PHP w dokumencie HTML

ZAGADNIENIA

- Jakie są cztery sposoby na umieszczenie skryptu w dokumencie HTML?
- Jakie rozszerzenie ma plik zawierający skrypt PHP?
- Jak wygląda komentarz w języku PHP?
- Jak wstawiać komentarze w języku PHP?

Kod PHP można zagnieździć w dokumencie HTML. Może to być pojedynczy skrypt lub kilka skryptów. Ważnym elementem jest wówczas zmiana rozszerzenia pliku z `.html` na `.php`. Jest to istotna informacja dla serwera, który rozpoczyna przetwarzanie zawartych w dokumencie skryptów. Najczęściej stosowanym sposobem umieszczania skryptu PHP w dokumencie jest:

```
<?php kod skryptu ?>
```

Pozwala on na rozpoznanie kodu PHP dla ustawień plików konfiguracyjnych serwera. Pozostałe sposoby bywają uzależnione od wprowadzenia zmian konfiguracyjnych:

```
<? kod skryptu ?>
<% kod skryptu %>
<script type="text/php"> kod skryptu  </script>
```

Pomiędzy otwarciem (`<?php`) i zamknięciem (`?>`) kodu PHP wprowadza się instrukcje zakończone znakiem średnika (`;`) (kod skryptu).

Wewnątrz kodu istnieje możliwość wprowadzania trzech rodzajów komentarzy. Pierwsze dwa są to komentarze jednoliniowe, które rozpoczynają się od podwójnego znaku slash (`//`) lub krzyżyka (`#`).

```
// komentarz jednoliniowy
# komentarz jednoliniowy
```

W przypadku, gdy istnieje konieczność ujęcia w komentarz większej części kodu, stosuje się komentarz wielowierszowy. Znakiem otwierającym komentarza wielowierszowego jest slash i gwiazdka (`/*`), a znakiem zamykającym – jego odwrotność (`*/`):

```
/*
komentarz
wieloliniowy
*/
```

SPRAWDŹ SWOJĄ WIEDZĘ

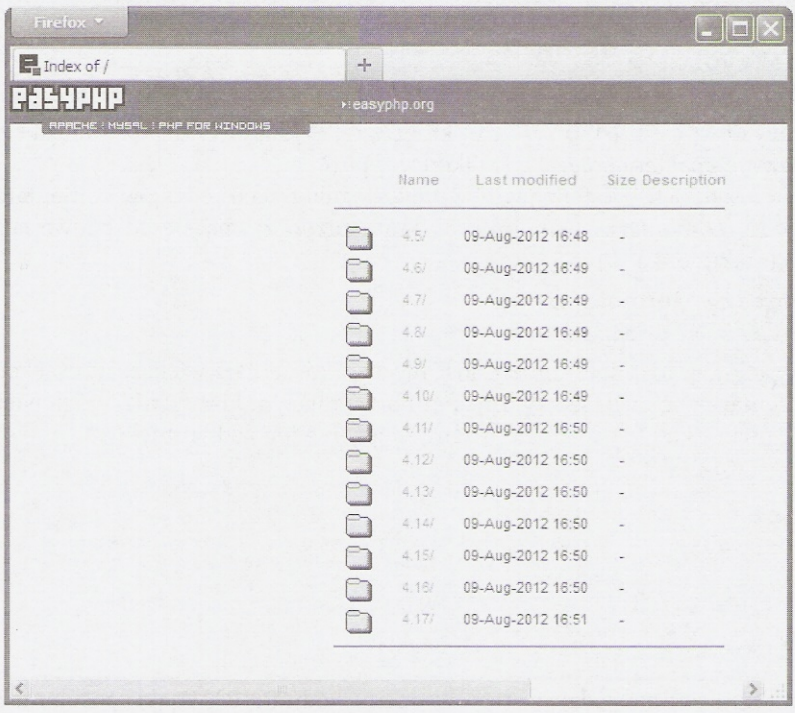
1. Jakie znaczniki odpowiadają za wprowadzenie kodu PHP? W jaki sposób można umieścić komentarz w kodzie PHP?

17 Instrukcje: print(), echo()

ZAGADNIENIA

- Za co odpowiada instrukcja print()?
- Za co odpowiada instrukcja echo()?
- Jak wyświetlić stronę z wykorzystaniem lokalnego serwera?
- Jak stworzyć pierwszy skrypt?
- Jak wyświetlać tekst na stronie za pomocą kodu PHP?

Tworząc skrypt PHP obsługiwany przez serwer lokalny (EasyPHP), należy pamiętać o zapisaniu go w odpowiednim folderze. W przypadku serwera EasyPHP jest to folder www znajdujący się w głównym katalogu serwera (np. C:\Program Files\EasyPHP\www). Wszystkie pliki oraz katalogi zawarte w tym folderze widoczne są po wprowadzeniu w przeglądarce adresu http://localhost/.



Rys. 17.1. http://localhost/ – widok zawartości folderu www

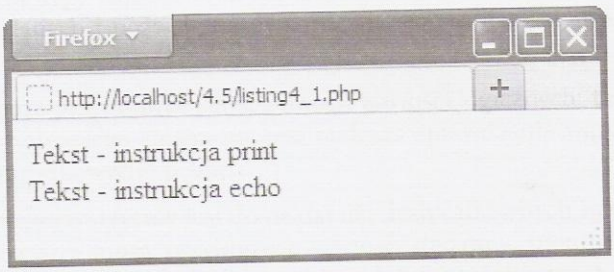
Po wybraniu konkretnego folderu pojawiają się dostępne pliki lub – jeżeli w folderze znajduje się plik o nazwie **index.php** – jego zawartość zostaje wyświetlona automatycznie.

Składnia języka PHP udostępnia dwie instrukcje pozwalające na wyświetlenie określonego tekstu na stronie. Pierwszą jest instrukcja **print** powodująca wyświetlenie treści umieszczonej w cudzysłowie. Kolejną instrukcją jest **echo**. Działa ona na takiej samej zasadzie jak instrukcja poprzednia.

Przykład z list. 17.1 przedstawia działanie obu instrukcji odpowiadających za wyświetlenie tekstu na stronie. W instrukcji **print** zagnieżdżono element składni dokumentu HTML. Znacznik **
** pozwala na wyświetlenie tekstu instrukcji **echo** w nowej linii.

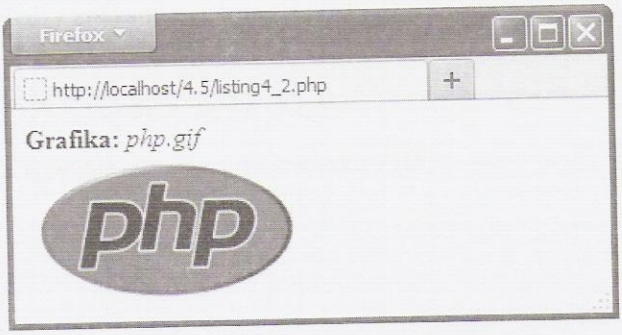
Listing 17.1

```
<?php
print "Tekst - instrukcja print<br>";
echo "Tekst - instrukcja echo";
?>
```



Rys. 17.2. Wynik działania skryptu z list. 17.1 – instrukcja **print** i **echo**

Przykład z list. 17.2 przedstawia zagnieżdżenie kodu HTML w języku PHP. W pierwszej instrukcji print wprowadzono trzy znaczniki HTML. Znacznik **<b>** jest odpowiedzialny za pogrubienie tekstu, znacznik **<i>** powoduje pochylenie tekstu, a znacznik **
** oznacza przejście do nowej linii. Dodatkowo wprowadzono znak kropki (**"."**), zwany inaczej operatorem konkatencji pozwalający na łączenie dowolnych ciągów znaków. Druga instrukcja print ma znacznik **img**, którego działanie związane jest z wyświetleniem obrazka (**php.gif**).



Rys. 17.3. Wynik działania skryptu z list. 17.2 – zagnieżdżenie kodu HTML

Listing 17.2

```
<?php
print "<b>Grafika:</b> "."<i>php.gif</i><br>";
print "<img src=\"php.gif\">";
?>
```

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Stosując zagnieżdżanie kodu HTML w języku PHP, wyświetl na stronie trzy obrazki za pomocą listy nienumerowanej, np.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Jak sprawdzić działanie skryptu na serwerze lokalnym?
2. Jakie instrukcje odpowiadają za wyświetlanie informacji w języku PHP?
3. Porównaj sposób wywoływania skryptów w języku PHP i JavaScript.

18 Typy danych

ZAGADNIENIA

- **Jakie typy danych występują w języku PHP?**
- **Czy PHP wymaga jawnej deklaracji typu danych?**
- **Jak stosować różne typy danych?**

Język PHP udostępnia osiem typów danych: jeden typ logiczny, dwa typy liczbowe, dwa typy specjalne, dwa typy złożone oraz typ łańcuchowy. PHP nie wymaga jawnego zadeklarowania typu danych i pozwala na dynamiczną jego zmianę w trakcie wykonywania skryptu.

Typ logiczny może przyjmować jedną z dwóch wartości logicznych: `true` (prawda) oraz `false` (fałsz). Najczęściej stosowany jest podczas sprawdzania warunków logicznych w instrukcjach warunkowych i pętlach.

Typ liczbowy wykorzystywany jest do deklaracji liczb całkowitych i zmiennoprzecinkowych. Liczby całkowite mogą przyjmować wartości dodatnie, ujemne oraz zero. Mogą zostać zapisane w notacji dziesiętnej, ósemkowej lub szesnastkowej. Należy pamiętać, że liczby całkowite i zmiennoprzecinkowe mają ograniczenia co do wielkości w zależności od platformy, na której działa PHP.

Typy specjalne to: wartość `NULL` oznaczająca, że do zmiennej nie przypisano żadnej wartości, oraz identyfikator zasobów, który ma odwołanie do zewnętrznego źródła danych (plik, baza danych).

Typ złożony określa dwa z najbardziej rozbudowanych typów danych. Pierwszy z nich to tablica, która służy do przechowywania większej ilości danych. Drugi typ to obiekty mające odpowiednie właściwości i własne metody (funkcje).

Typ łańcuchowy odpowiada za przechowywanie ciągów znaków (łańcuchów). Ciąg należy umieścić pomiędzy znakami apostrofów lub cudzysłowów.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Scharakteryzuj typy danych dostępne w języku PHP.

19 Zmienne i stałe

ZAGADNIENIA

- Co to jest zmienna?
- Co to jest stała?
- Jakie operatory można zastosować w języku PHP?
- Jak deklarować i wykonywać operacje na zmiennych?
- Jak deklarować i wykonywać operacje na stałych?

Zmienna jest to określony obszar pamięci umożliwiający przechowywanie różnych wartości, do których jest stały dostęp i które mogą ulec zmianie podczas wykonywania skryptu. W języku PHP każda zmienna rozpoczyna się znakiem dolara ($), po którym podawana jest nazwa zmiennej.

```
$nazwa_zmiennej;
```

Nazwa zmiennej jest dowolnym ciągiem znaków składających się z liter, liczb oraz dolnego podkreślenia (_). Należy pamiętać, że nazwa zmiennej nie może zaczynać się od cyfry. Język PHP rozróżnia wielkość liter, więc np. zmienne o nazwach `$zmienna1` i `$ZMIENNA1` to dwie różne zmienne.

Stała to również obszar w pamięci przechowujący pewną wartość, która jednak nie może ulec zmianie podczas wykonywania skryptu. Stałe definiowane są w języku PHP za pomocą funkcji `define()`. Przyjmuje ona dwa parametry: nazwę zmiennej i wartość, jaka została do niej przypisana:

```
define("nazwa_stałej", "wartość");
```

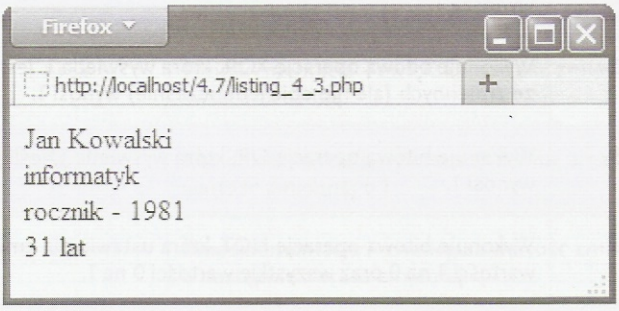
Nazwa stałej jest dowolnym ciągiem znaków składającym się z liter, liczb i dolnego podkreślenia. Należy pamiętać, że nazwa stałej nie może zaczynać się od cyfry. Język PHP rozróżnia wielkość liter, więc np. stałe o nazwach `procent` i `Procent` to dwie różne stałe.

Przykład z list. 19.1 przedstawia deklarację dwóch wartości stałych oraz dwóch zmiennych. W obu przypadkach zastosowano zarówno typ danych łańcuchowy, jak i liczbowy. Wszystkie wartości zostały wyświetlone na stronie za pomocą instrukcji `print()`. Do połączenia kilku ciągów lub ciągu i zmiennej zastosowano kropkę –. – operator ciągu.

Listing 19.1

```
<?php
define("imie","Jan Kowalski");
define("rok",1981);
$zawod="informatyk";
$wiek=31;
print(imie."<br>");
```

```
print($zawod."<br>");
print("rocznik - ".rok."<br>");
print($wiek." lat <br>");
?>
```



Rys. 19.1. Wynik działania skryptu z list. 19.1 – deklaracja zmiennej i stałej

Na zmiennych i stałych mogą być wykonywane operacje z wykorzystaniem różnych operatorów. Język PHP udostępnia następujące operatory: arytmetyczne, logiczne, bitowe, przypisania, porównania, inkrementacji, dekrementacji oraz operator obsługi błędów i operator ciągu.

Operatory arytmetyczne wykorzystywane są do wykonywania na zmiennych i stałych podstawowych operacji matematycznych. Operatory arytmetyczne przedstawiono w tabeli 19.1.

Tabela 19.1. Operatory arytmetyczne

Symbol	Składnia	Opis
+	$x+$y	Wykonuje operację dodawania.
-	$x-$y	Wykonuje operację odejmowania.
*	$x*$y	Wykonuje operację mnożenia.
/	$x/$y	Wykonuje operację dzielenia.
%	$x%$y	Zwraca resztę z dzielenia pierwszej zmiennej przez drugą (dzielenie modulo).

Operatory bitowe stosowane są do wykonywania na bitach operacji algebry logicznej. Dostępne operatory przedstawiono w tabeli 19.2.

Tabela 19.2. Operatory bitowe

Symbol	Składnia	Opis
&	$x&$y	Wykonuje bitową operację AND, która wyświetla 1, jeśli obie zmienne wynoszą 1.
^	$x^$y	Wykonuje bitową operację XOR, która wyświetla 1, jeśli jedna ze zmiennych (ale nie obie jednocześnie) wynosi 1.
|	$x\|$y	Wykonuje bitową operację OR, która wyświetla 1, jeśli jedna ze zmiennych wynosi 1.
~	~$x	Wykonuje bitową operację NOT, która ustawia dla zmiennej wszystkie wartości 1 na 0 oraz wszystkie wartości 0 na 1.
<<	$x<<$y	Wykonuje przesunięcie bitów w lewo o podaną liczbę miejsc.
>>	$x>>$y	Wykonuje przesunięcie bitów w prawo o podaną liczbę miejsc.

Podstawowym **operatorem przypisania** jest znak = odpowiadający za przypisanie wartości argumentu prawostronnego argumentowi lewostronnemu. Argumentem prawostronnym może być zmienna, stała lub wyrażenie, natomiast argument lewostronny stanowi zmienna, która przyjmie nową wartość. Język PHP oferuje dodatkowo wiele operatorów łączonych, zaprezentowanych w tabeli 19.3.

Tabela 19.3. Operatory przypisania

Symbol	Składnia	Opis
=	$x=$y	Przypisuje wartość $y do zmiennej $x.
+=	$x+=$y	Wykonuje przypisanie $x=$x+$y.
-=	$x-=$y	Wykonuje przypisanie $x=$x-$y.
*=	$x*=$y	Wykonuje przypisanie $x=$x*$y.
/=	$x/=$y	Wykonuje przypisanie $x=$x/$y.
%=	$x%=$y	Wykonuje przypisanie $x=$x%$y.

Inkrementacja i **dekrementacja** to operatory stosowane odpowiednio do zwiększania i zmniejszania wartości danej zmiennej o 1. Stosowane wersje obu operatorów przedstawiono w tabeli 19.4.

Tabela 19.4. Operator inkrementacji i dekrementacji

Symbol	Składnia	Opis
++	$x++	Postinkrementacja – zwraca zmienną, a następnie zwiększa wartość zmiennej o 1.
++	++$x	Preinkrementacja – zwiększa wartość zmiennej o 1, a następnie zwraca zmienną.
--	$x--	Postdekrementacja – zwraca zmienną, a następnie zmniejsza wartość zmiennej o 1.
--	--$x	Predekrementacja – zmniejsza wartość zmiennej o 1, a następnie zwraca zmienną.

Operatory porównania wykorzystywane są do porównania dwóch argumentów. Jeżeli na wyjściu otrzymamy wartość **`true`**, oznacza to, że warunek został spełniony, w przeciwnym wypadku zwrócona zostanie wartość **`false`**.

Tabela 19.5. Operatory porównania

Symbol	Składnia	Opis
!=	$x!=$y	Zwraca **`true`**, jeśli zmienne nie są równe.
!==	$x!==$y	Zwraca **`true`**, jeśli zmienne nie są równe.
<	$x<$y	Zwraca **`true`**, jeśli pierwsza zmienna jest mniejsza niż druga.
<=	$x<=$y	Zwraca **`true`**, jeśli pierwsza zmienna jest mniejsza niż druga lub jej równa.
==	$x==$y	Zwraca **`true`**, jeśli zmienne są równe.
===	$x===$y	Zwraca **`true`**, jeśli zmienne są równe.
>	$x>$y	Zwraca **`true`**, jeśli pierwsza zmienna jest większa niż druga.
>=	$x>=$y	Zwraca **`true`**, jeśli pierwsza zmienna jest większa niż druga lub jej równa.

Operatory logiczne: koniunkcja (**`&&`**), alternatywa (**`||`**) oraz negacja (**`!`**) zwracają wartość **`true`** (prawda) lub **`false`** (fałsz) według zależności przedstawionych w tabeli 19.6.

Tabela 19.6. Operatory logiczne

Symbol	Składnia	Opis
&&	$x && $y	Operator logiczny AND zwraca **true**, jeśli obie zmienne są prawdziwe (**true**).
||	$x || $y	Operator logiczny OR zwraca **true**, jeśli co najmniej jedna ze zmiennych jest prawdziwa (**true**).
!	!$x	Ten operator logiczny neguje wyrażenie.

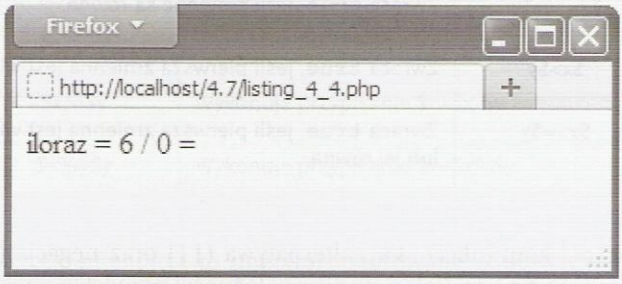
Listing 19.2

```
<?php
$dzielna=6;
$dzielnik=0;
@print(„iloraz = ".$dzielna." / ".$dzielnik." = ".$dzielna
/$dzielnik);
?>
```

W języku PHP występuje dodatkowo **operator kontroli błędów** oznaczany znakiem tzw. małpy @. Zastosowanie go przed wybranym wyrażeniem spowoduje, że w wypadku wygenerowania błędu przez to wyrażenie na stronie nie pojawi się ostrzeżenie ani żadna informacja o błędzie. Operator ten można umieszczać przed zmiennymi, stałymi, instrukcjami **include()** oraz wywołaniami funkcji. Nie wolno stosować go przed definicją funkcji i klasy oraz przed strukturami kontrolnymi, takimi jak **if**, **while** czy **foreach**.

Operatorem ciągu (konkatencji, łączenia) jest kropka. Operator ten odpowiada za łączenie kilku ciągów w jeden. Może również połączyć ciąg ze zmienną lub z wyrażeniem.

Listing 19.2 przedstawia przykład zastosowania operatora błędu. Znak „małpy" został postawiony przed instrukcją **print()**, której zadaniem jest wypisanie wyniku dzielenia dwóch zmiennych. Zmiennej **$dzielnik** przypisano wartość 0. W takim wypadku PHP wyświetliłoby ostrzeżenie związane z dzieleniem przez 0 (Warning: **Division by zero**). Operator kontroli błędu ukryje ostrzeżenie. Wyświetli się jedynie część informacji bez podania wyniku dzielenia (rys. 19.2).



Rys. 19.2. Wynik działania skryptu z list. 19.2 – operator kontroli błędów

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Napisz skrypt przeliczający walutę z euro na złotówki i na odwrót. Aktualny kurs ustaw jako wartość stałą.
2. Przeanalizuj poniższy skrypt i przedstaw, w jaki sposób zmieniają się wartości zmiennych x, y i z podczas kolejnych kroków programu.

```
<?php
$x=2;
$y=3;
$z=4;
$x++;
$y--;
$z+=$y;
--$z;
$x*=$z;
print($x." ".$y." ".$z);
?>
```

SPRAWDŹ SWOJĄ WIEDZĘ

1. Scharakteryzuj pojęcie zmiennej.
2. Wymień operatory języka PHP.
3. Jak zadeklarować wartość stałą w PHP?
4. W jakim celu stosowane są wartości stałe w skryptach?

20 Instrukcje warunkowe

ZAGADNIENIA

- Jak wygląda ogólny zapis instrukcji warunkowej?
- Co to jest instrukcja wyboru?
- Jak stosować instrukcję warunkową?
- Jak wykorzystywać instrukcje wyboru?

Instrukcja warunkowa jest jednym z najczęściej stosowanych wyrażeń programistycznych. Decyduje, który z fragmentów skryptu zostanie wykonany w zależności od spełnienia określonego warunku. Zapis instrukcji warunkowej pozwala na wykonanie **instrukcji**, gdy warunek przyjmie wartość **true**:

```
if (warunek) {
instrukcje;
}
```

Forma bardziej rozbudowana instrukcji warunkowej ma dodatkowo element **else**. W tym wypadku **instrukcja_1** zostanie wykonana, gdy warunek przyjmie wartość **true**. W przeciwnym wypadku, czyli wtedy, gdy warunek przyjmie wartość **false**, zostanie wykonana **instrukcja_2**.

```
if (warunek) {
instrukcja_1;
}
else {
instrukcja_2;
}
```

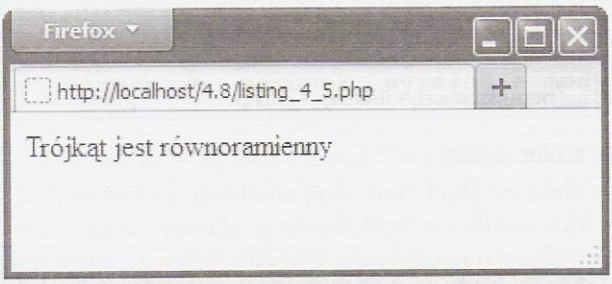
Ostatnia forma instrukcji warunkowej składa się z kilku następujących po sobie warunków. W jej skład wchodzi element **elseif(warunek)**, który może pojawić się częściej niż raz:

```
if (warunek) {
instrukcja_1;
}
elseif(warunek_2) {
instrukcja_2;
}
else {
instrukcja_3;
}
```

Przykład z list. 20.1 przedstawia zastosowanie rozbudowanej instrukcji warunkowej do sprawdzenia na podstawie trzech długości boków (**$a**, **$b**, **$c**), czy dany trójkąt jest równoboczny, równoramienny, czy różnoboczny.

Listing 20.1

```
<?php
$a=3;
$b=5;
$c=5;
if($a==$b&&$b==$c) {
print("Trójkąt o bokach".$a.",".$b.",".$c."jest równoboczny");
}
elseif($a==$b||$b==$c||$a==$c) {
print("Trójkąt o bokach".$a.",".$b.",".$c."jest równoramienny");
}
else {
print("Trójkąt o bokach".$a.",".$b.",".$c."jest różnoboczny");
}
?>
```



Rys. 20.1. Wynik działania skryptu z list. 20.1 – instrukcja warunkowa

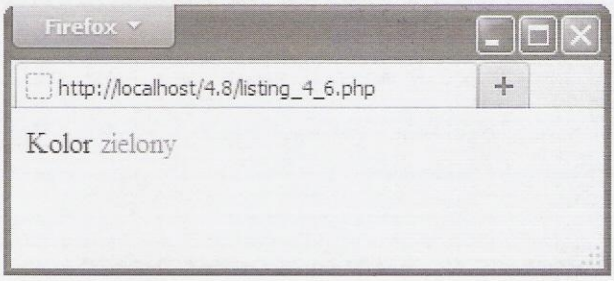
Instrukcja wyboru **switch** działa podobnie jak najbardziej rozbudowana wersja instrukcji warunkowej. Zasada działania tej instrukcji polega na tym, że na początku ustalana jest wartość wyrażenia, które porównywane jest z wartościami wprowadzonymi dla każdego przypadku (**case**). Jeżeli wartość jest równa wyrażeniu, następuje wykonywanie instrukcji dla każdego kolejnego przypadku, aż do napotkania instrukcji **break**, która powoduje opuszczenie instrukcji **switch**. Instrukcje przypisane do etykiety **default** zostaną wykonane, jeżeli żadna z wartości nie odpowiada założonemu wyrażeniu.

```
switch(wyrażenie){
case "wartość_1":
instrukcja_1;
break;
case "wartość_2":
instrukcja_2;
break;
default:
instrukcja;
}
```

Przykład z list. 20.2 przedstawia instrukcję wyboru, która jako wyrażenie przyjmuje wartość zmiennej łańcuchowej **$kolor**. W zależności od tego, jaką wartość przyjmie zmienna, w tym przypadku **zielony** (lub **czerwony**), na stronie zostanie wypisana treść w danym kolorze lub informacja **Podano błędny kolor!**

Listing 20.2

```
<?php
$kolor="zielony";
switch($kolor){
case "czerwony":
print("Kolor <font color=\"red\">czerwony</font>");
break;
case "zielony":
print("Kolor <font color=\"green\">zielony</font>");
break;
default:
print("Podano błędny kolor!");
}
?>
```



Rys. 20.2. Wynik działania skryptu z list. 20.2 – wyrażenie warunkowe

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Napisz skrypt sprawdzający, czy wprowadzona przez użytkownika wartość jest podzielna przez 2, 3 i 5 (rozpatrz wszystkie przypadki).

SPRAWDŹ SWOJĄ WIEDZĘ

1. Do czego wykorzystywana jest instrukcja warunkowa?
2. Podaj ogólny zapis instrukcji wyboru.
3. Przedstaw schemat blokowy instrukcji wyboru. Scharakteryzuj elementy blokowe zawarte w schemacie.

21 Pętle

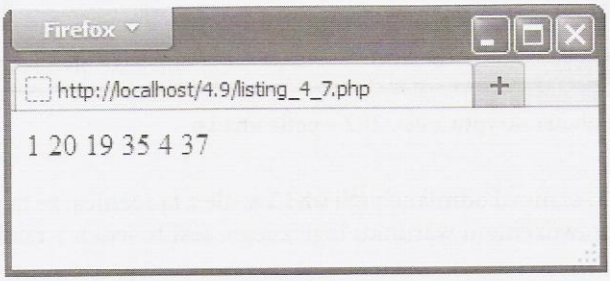
ZAGADNIENIA

- Jakie zadanie mają pętle?
- Jakie pętle dostępne są w języku PHP?
- Jak stosować pętle w skrypcie?
- Jak stosować instrukcję warunkową wewnątrz pętli?

Pętla to konstrukcja programistyczna pozwalająca na wykonanie wybranych instrukcji określoną liczbę razy. Najczęściej stosowaną pętlą w języku PHP jest pętla **for**. Przyjmuje ona trzy parametry: **wyrażenie początkowe** – ustala wartość dla zmiennej kontrolującej pętlę, **warunek** – wyrażenie logiczne warunkujące zakończenie działania pętli oraz **wyrażenie modyfikujące** – zmiana zmiennej kontrolującej pętle:

```
for (wyrażenie początkowe; warunek; wyrażenie modyfikujące){
instrukcje;
}
```

Przykład z list. 21.1 prezentuje działanie pętli **for**. Pętla zostanie wykona sześć razy. Jej zadaniem jest wyświetlenie sześciu losowych liczb z zakresu od 1 do 49. W tym celu wykorzystano funkcję **rand()**, której zadaniem jest wygenerowanie pseudolosowej liczby stałoprzecinkowej. Funkcja może przyjmować dwa parametry określające zakres liczbowy, z jakiego będą losowane liczby. Istnieje możliwość zastosowania funkcji bezparametrowej. Wówczas przyjmie ona zakres liczbowy od 0 do RAND_MAX.



Rys. 21.1. Wynik działania skryptu z list. 21.1 – pętla **for**

Listing 21.1

```
<?php
for($i=0;$i<6;$i++) {
```

```
print(rand(1,49)." ");
}
?>
```

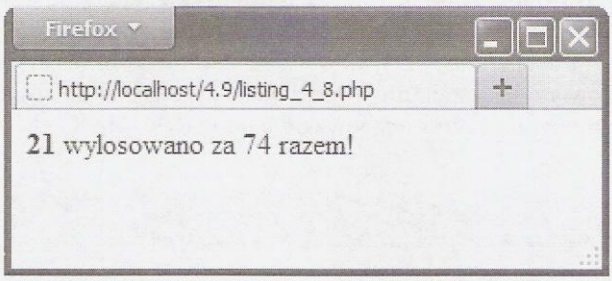
Innym rodzajem pętli jest pętla **while**. Jest ona wykonywana, dopóki warunek logiczny jest spełniony. Jest to najlepsza pętla, jeżeli chcemy wykonywać daną operację do czasu zajścia określonego warunku. Dlatego idealnie nadaje się do odczytywania plików. Pętla wówczas jest wykonywana tak długo, aż napotka koniec pliku.

```
while (warunek){
instrukcje;
}
```

Przykład z list. 21.2 przedstawia działanie pętli **while**. Warunkiem zakończenia działania pętli jest wylosowanie przez funkcję losującą **rand()** liczby 21. Wewnątrz pętli umieszczono licznik. Zlicza on liczbę wykonanych pętli. W rezultacie podawana jest informacja, za którym razem została wylosowana liczba 21. W tym przypadku przy każdym uruchomieniu skryptu liczba wykonanych pętli będzie inna.

Listing 21.2

```
<?php
$x=1;
while(rand(1,100)!=21) {
$x++;
}
print("<b>21</b> wylosowano za ".$x." razem!");
?>
```



Rys. 21.2. Wynik działania skryptu z list. 21.2 – pętla **while**

Pętla **do/while** stanowi odmianę pętli **while**, ale z tą różnicą, że ta pętla jest wykonywana raz przed sprawdzeniem warunku logicznego. Jest to jeden z rzadziej stosowanych rodzajów pętli.

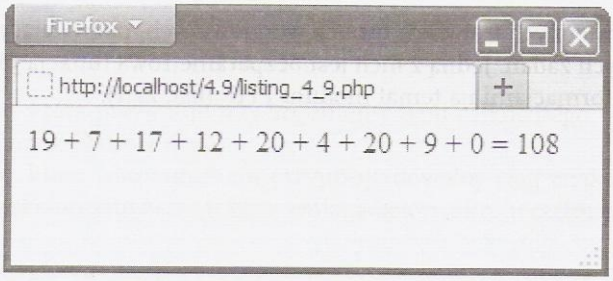
```
do {
instrukcje;
}
while(warunek)
```

Przykład z list. 21.3 przedstawia działanie pętli **do/while**. Kolejne wykonania pętli powodują sumowanie się wylosowanych liczb z zakresu od 1 do 20. Pętla będzie wykonywa-

na do czasu przekroczenia przez zmienną **$suma** wartości 100. Końcowym efektem jest wypisanie w oknie przeglądarki sumy poszczególnych wylosowanych liczb oraz wyniku końcowego.

Listing 21.3

```
<?php
$suma=0;
do {
$x=rand(1,20);
$suma=$suma+$x;
print($x." + ");
}
while($suma<100);
print("0 = ".$suma);
?>
```



Rys. 21.3. Wynik działania skryptu z list. 21.3 – pętla **do/while**

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Stosując pętle i jedną instrukcję **print("*")**, wyświetl w przeglądarce następujące struktury:
   a) linia składająca się z dziesięciu *:

      ****

   b) trójkąt o podstawie sześciu *:

      *
      **
      ***
      ****
      *****
      ******

SPRAWDŹ SWOJĄ WIEDZĘ

1. W jakim celu stosowane są pętle w językach programowania (skryptach)?
2. Przedstaw schemat blokowy, strukturę i opisz elementy pętli **for**.
3. Na czym polega różnica między pętlą **while** a **do/while**?

22 Funkcje

ZAGADNIENIA

- Co to jest funkcja?
- Co to są argumenty funkcji?
- Jak definiować funkcję?
- Jak wywoływać funkcję?

W języku PHP dostępnych jest wiele funkcji wbudowanych, pozwalających na wykonanie szeregu przydatnych zadań. Jedną z nich jest bezparametrowa funkcja **phpinfo()** generująca stronę z informacjami na temat modułu PHP (list. 22.1).

Listing 22.1

```
<?php
phpinfo();
?>
```

Język PHP umożliwia również tworzenie własnych funkcji. **Funkcja** stanowi fragment kodu, który może zostać kilkakrotnie wykonany poprzez wywołanie danej funkcji. Definiując funkcję, należy wprowadzić słowo kluczowe **function**, określić nazwę funkcji oraz podać jej argumenty lub pozostawić funkcję bezparametrową. W nawiasach klamrowych podajemy instrukcje, które wykona funkcja.

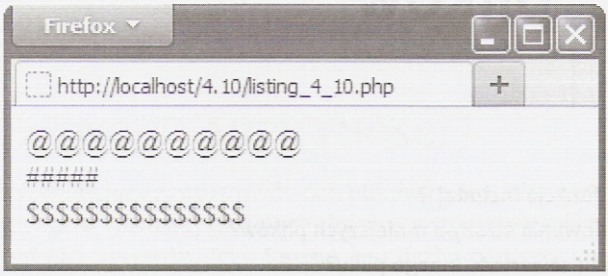
```
function nazwa_funkcji(agrument1, argument2, ... ,argument n){
instrukcje;
}
```

Przykład z list 22.2 prezentuje działanie funkcji. Funkcja ma dwa argumenty: **$znak** oraz **$dlugosc**. Jej zadaniem jest wyświetlenie w oknie przeglądarki łańcucha zbudowanego z symboli przypisanych do argumentu **$znak**. Długość łańcucha określa drugi argument funkcji. Funkcja została wywołana trzy razy, za każdym razem z innymi argumentami.

Listing 22.2

```
<?php
function lancuch($znak,$dlugosc) {
for($i=0;$i<$dlugosc;$i++)
print($znak);
print("<br>");
}
```

```
lancuch("@",10);
lancuch("#",5);
lancuch("$",15);
?>
```



Rys. 22.1. Wynik działania skryptu z list. 22.2 – funkcja

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Napisz funkcję, która przyjmuje trzy argumenty typu całkowitego. Zadaniem funkcji jest wypisanie największej z liczb.
2. Napisz funkcję, która jako argument przyjmuje dowolny ciąg znaków ($tekst). Zadaniem funkcji jest sformatowanie tekstu: zmiana koloru i kroju czcionki oraz pochylenie.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Podaj przykład funkcji wbudowanej.
2. Jak wywołać funkcję z argumentem? Podaj przykład.
3. Podaj przykład funkcji wbudowanej w języki PHP. Przedstaw jej odpowiednik w języku JavaScript.

23 Instrukcje dołączania plików

ZAGADNIENIA

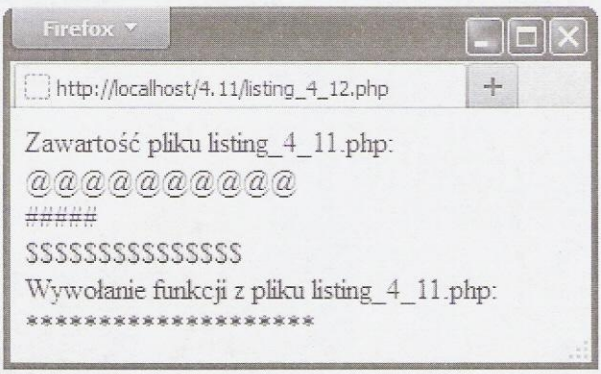
- Za co odpowiada funkcja include()?
- Jakie są zalety budowania strony z mniejszych plików?
- Jak włączyć do kodu zawartość innego pliku?

W języku PHP dostępne są dwie instrukcje pozwalające na włączenie do kodu skryptu zawartości innego pliku zawierającego kod PHP. Instrukcje **include()** oraz **require()** przyjmują jeden argument, którym jest nazwa pliku lub cała ścieżka dostępu do pliku, mającego znajdować się na innym serwerze.

```
include(nazwa_pliku.php);
require(nazwa_pliku.php);
```

Obie instrukcje wywołują się podobnie, ale występują między nimi różnice. Przy wywoływaniu instrukcji **require()** nazwa pliku nie może zostać pobrana ze zmiennej, **include()** zaś daje taką możliwość. W wypadku gdy plik nie zostanie odnaleziony, **require()** wygeneruje błąd **Fatal error** i zatrzyma skrypt. Natomiast **include()** jedynie wyświetli ostrzeżenie.

Przykład z list 23.1 przedstawia zastosowanie funkcji **include()**, której zadaniem jest wywołanie kodu zawartego w pliku listing_4_11.php. Plik ten zawiera funkcję wyświetlającą na stronie łańcuch złożony z odpowiednich znaków i o określonej długości. Oba parametry określają argumenty funkcji **lancuch($znak,$dlugosc)**. Dodatkowo po dołączeniu pliku istnieje możliwość wywołania zawartej w nim funkcji bezpośrednio z głównego kodu skryptu – **lancuch("*",20)**.



Rys. 23.1. Wynik działania skryptu z list. 23.1 – instrukcja **include()**

Listing 23.1

```
<?php
print("Zawartość pliku listing_4_11.php: <br>");
include("listing_4_11.php");
print("Wywołanie funkcji z pliku listing_4_11.php: <br>");
lancuch("*",20);
?>
```

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Napisz skrypt wyświetlający zawartość dwóch plików **.php**. Pierwszy plik zawiera funkcję wyświetlającą sumę (**$suma**) dziesięciu losowo wygenerowanych liczb z przedziału od 1 do 15. Drugi plik sprawdza, czy **$suma** jest liczbą parzystą i wyświetla stosowną informację.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Jakie są sposoby na włączenie do kodu zawartości innego pliku **.php**?
2. Jaka jest różnica pomiędzy funkcjami **include()** a **require()**?

24 Tablice

ZAGADNIENIA

- Co to jest tablica?
- Jak wygląda tablica asocjacyjna?
- Jak wprowadzać dane do tablicy jednowymiarowej i wyprowadzać je z niej?
- Jak wprowadzać dane do tablicy asocjacyjnej i wyprowadzać je z niej?

Tablica jest zmienną przechowującą zbiór określonych wartości (elementów). Do każdej z wartości można się odwołać, używając odpowiedniego indeksu. Indeks to kolejny numer elementu w tablicy, począwszy od 0.

Tablica jest zmienną strukturalną i wobec jej nazwy obowiązują te same reguły co dla zmiennych prostych, czyli przed nazwą umieszczony zostaje znak **$**. Pierwszy sposób deklaracji tablicy to użycie słowa kluczowego **array**:

```
$nazwa_tablicy=array(element_1, element_2, ... , element_n);
```

Innym sposobem jest deklaracja z użyciem nawiasów kwadratowych. Jawne określenie indeksu nie jest konieczne. Poszczególne elementy zostaną wpisane do tablicy kolejno od indeksu 0. Elementami tablicy mogą być liczby lub łańcuchy znaków.

```
$nazwa_tablicy[]=element_1;
$nazwa_tablicy[]=element_2;
...
$nazwa_tablicy[]=element_n;
```

Przykład z list. 24.1 przedstawia jeden ze sposobów uzupełnienia tablicy jednowymiarowej. Do **$tablica[]** wprowadzono dziesięć liczb wygenerowanych przez funkcję losującą **rand()** z przedziału liczbowego od 1 do 100. Następnie wprowadzono dwie funkcje losujące, które przyjmą tablicę jako argument. Funkcja **sort()** sortuje tablicę rosnąco, a funkcja **rsort()** – malejąco.

Listing 24.1

```
<?php
for($i=0;$i<10;$i++) {
$tablica[$i]=rand(1,100);
print($tablica[$i]." ");
}
sort($tablica);
print("<br> Tablica posortowana rosnąco: <br>");
for($i=0;$i<10;$i++) {
print($tablica[$i]." ");
```

```
}
rsort($tablica);
print("<br> Tablica posortowana malejąco: <br>");
for($i=0;$i<10;$i++) {
print($tablica[$i]." ");
}
?>
```

Rys. 24.1. Wynik działania skryptu z list. 24.1 – tablica jednowymiarowa

Tablica asocjacyjna jest specyficzną odmianą tablicy, która zamiast zwykłych indeksów numerycznych stosuje indeksy znakowe zwane kluczami.

Jeden ze sposób deklaracji tablicy asocjacyjnej to użycie słowa **array** oraz kluczy pełniących podobną rolę jak indeksy w zwykłej tablicy:

```
$nazwa_tablicy=array(klucz_1=>element_1, ... , klucz_n=>elemen-
t_n);
```

Kolejnym sposobem jest użycie nawiasów kwadratowych, wewnątrz których należy umieścić klucz.

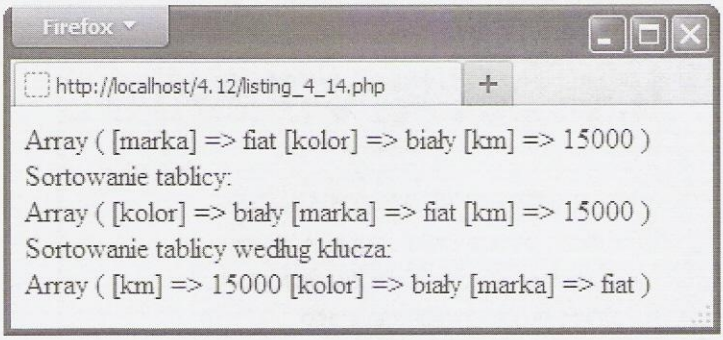
```
$nazwa_tablicy[klucz_1]=element_1;
...
$nazwa_tablicy[klucz_n]=element_n;
```

Przykład z list. 24.2 prezentuje utworzenie prostej tablicy asocjacyjnej **$auta**. Zawartość tablicy wyświetlona została za pomocą funkcji **print_r()**, która wyświetla informacje w sposób zrozumiały dla użytkownika. PHP udostępnia funkcje pozwalające na sortowanie tablicy asocjacyjnej. Funkcja **asort()** sortuje tablicę, zachowując przypisanie kluczy do wartości. Funkcja **ksort()** sortuje tablicę według kluczy.

Listing 24.2.

```
<?php
$auta["marka"]="fiat";
$auta["kolor"]="biały";
$auta["km"]=15000;
print_r($auta);
asort($auta);
print("<br>Sortowanie tablicy:<br>");
```

```
print_r($auta);
ksort($auta);
print("<br>Sortowanie tablicy według klucza:<br>");
print_r($auta);
?>
```



Rys. 24.2. Wynik działania skryptu z list. 24.2 – tablica asocjacyjna

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Napisz funkcję, która wpisuje do tablicy liczby z zakresu określonego przez dwa argumenty funkcji oraz wyświetla sumę elementów tablicy.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Czym są tablica i indeks?
2. Co to jest tablica asocjacyjna i jakie zadanie ma klucz?

25 Programowanie obiektowe

ZAGADNIENIA

- Czym są klasa, obiekt i metody?
- Za co odpowiada konstruktor?
- Na czym polega dziedziczenie?
- Jak definiować klasę?
- Jak tworzyć obiekty?

Poznane do tej pory elementy programowania w języku PHP związane były głównie z programowaniem strukturalnym. W wersji 5. PHP zdecydowanie rozbudowano możliwości programowania obiektowego. Jest to obecnie najpopularniejsza technika programowania.

Z programowaniem obiektowym ściśle związane są dwa pojęcia: klasa oraz obiekt. **Klasa** to specjalna konstrukcja programistyczna zawierająca właściwości obiektu oraz metody (funkcje), które operują na danym obiekcie. **Obiekt** stanowi pewne odwzorowanie rzeczywistości. Może to być dowolna rzecz, która ma cechy charakterystyczne odpowiadające za opisanie właściwości obiektu. Każdy obiekt może być zbudowany z dodatkowych obiektów oraz każda klasa może mieć kilka obiektów.

Stworzenie obiektu rozpoczyna się od stworzenia klasy. Podstawowa definicja klasy przedstawia się następująco:

```
class nazwa_klasy {
//instrukcje klasy
}
```

Klasa rozpoczyna się od słowa kluczowego **class**, po którym należy wprowadzić nazwę klasy oraz w nawiasie klamrowym odpowiednie instrukcje klasy. Zwyczajowo przyjęło się zapisywać nazwę klasy wielką literą. Dodatkowo wewnątrz klasy należy poprzedzić metody i właściwości obiektu specyfikatorem dostępu: **private** – składowe są widoczne tylko w obrębie metody danej klasy, **public** – składowe są widoczne w całym skrypcie oraz **protected** – składowe są dostępne w danej klasie i klasach potomnych.

Po utworzeniu klasy można przystąpić do tworzenia obiektu. Należy rozpocząć od wprowadzenia słowa kluczowego **new** oraz nazwy klasy opatrzonej nawiasami okrągłymi:

```
new nazwa_klasy();
```

Następnie utworzony obiekt należy przypisać do zmiennej, aby można było z niego swobodnie korzystać:

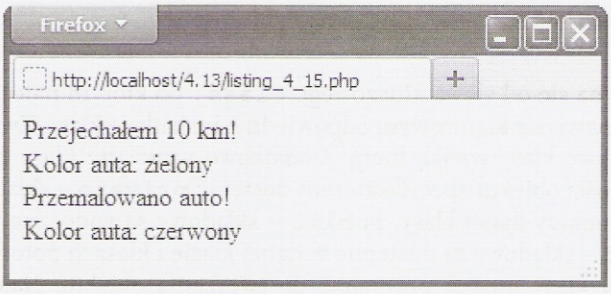
```
$nazwa_zmiennej = new nazwa_klasy();
```

Przykład z list. 25.1 przedstawia deklarację klasy **Pojazd** i utworzenie obiektu tej klasy, który został przypisany do zmiennej **$auto**. Klasa posiada trzy pola obiektu **$auto**,

mające specyfikator dostępu **public**. Do każdego pola została przypisana wartość przez podanie nazwy obiektu, znaku strzałki (->) oraz nazwy pola niepoprzedzonej znakiem **$**. Klasa ma dwie metody (funkcje): **jazda()** oraz **malowanie()**. Zadaniem pierwszej jest jedynie wypisanie informacji. Metoda druga powoduje zmianę wartości pola **$kolor**. Odbywa się to za pomocą zmiennej **$this**, która wewnątrz metody danego obiektu jest jego synonimem.

Listing 25.1

```
<?php
class Pojazd {
public $marka;
public $kolor;
public $rocznik;
   function jazda() {
      print("Przejechałem 10 km! <br>");
   }
   function malowanie() {
      $this->kolor="czerwony";
      print("<br> Przemalowano auto! <br>");
   }   }
$auto=new Pojazd();
$auto->marka="BMW";
$auto->kolor="zielony";
$auto->rocznik=2011;
$auto->jazda();
print("Kolor auta: ".$auto->kolor);
$auto->malowanie();
print("Kolor auta: ".$auto->kolor);
?>
```



Rys. 25.1. Wynik działania skryptu z list. 25.1 – klasa i obiekt

Każda klasa ma dwie charakterystyczne metody, określone jako: konstruktor i destruktor. **Konstruktor** to metoda wywoływana automatycznie po utworzeniu obiektu danej klasy. Jego zadania to inicjalizacja pól oraz czynności związane z poprawnym działaniem tworzonego obiektu. **Destruktor** wywoływany jest w momencie usuwania obiektu z pamięci. Metoda ta odpowiada za zakończenie wszystkich prac powiązanych z obiektem: zamknięcie plików, zakończenie pracy z bazą danych, zwolnienie zasobów. Definicję konstruktora i destruktora należy wprowadzić wewnątrz klasy:

```
class nazwa_klasy {
   function __construct(){
      //instrukcje konstruktora
   }
   function __destruct(){
      //instrukcje konstruktora
   }
//instrukcje klasy
}
```

Definiowanie konstruktora i destruktora nie jest konieczne. Interpreter PHP dla każdej klasy generuje automatycznie konstruktora i destruktora pustego.

Kolejnym ważnym zagadnieniem związanym z programowaniem obiektowym jest **dziedziczenie**, które pozwala na wykorzystanie już istniejących klas. Klasę dziedziczoną nazywa się nadklasą (klasą nadrzędną), a klasę dziedziczącą – podklasą (klasą podrzędną). Pozwala to na zbudowanie hierarchicznego układu klas. W języku PHP dziedziczenie jest oznaczane za pomocą słowa kluczowego **extends** i przedstawia się następująco:

```
class klasa_nadrzedna {
//instrukcje klasy nadrzędnej
}
class klasa_podrzedna extends klasa_nadrzedna {
//instrukcje klasy podrzędnej
}
```

Klasa podrzędna ma dostęp do wszystkich elementów klasy nadrzędnej oznaczonych jako **public** i **protected**. Natomiast elementy oznaczone jako **private** są niedostępne w klasie podrzędnej.

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Zdefiniuj klasę **Osoba**, która ma trzy pola: **$imię**, **$nazwisko** i **$rok_urodzenia** oraz dwie metody: **wypisz()** – wyświetlającą wszystkie dane w przeglądarce oraz **osiemnascie()** – sprawdzającą, czy dana osoba jest pełnoletnia.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Czym są klasa i obiekt?
2. Czym są metody klasy?
3. Za co odpowiada dziedziczenie?

26 Obsługa wyjątków

ZAGADNIENIA

- Co to jest wyjątek?
- Jakie metody wykorzystać do obsługi wyjątków?
- Jak przechwytywać wyjątki występujące w skrypcie?
- Jak obsługiwać sytuacje wyjątkowe?

Wyjątek to zaistnienie pewnej sytuacji podczas działania skryptu, która może spowodować jego przerwanie lub błędne wykonanie. Warto się przed takimi zdarzeniami zabezpieczyć, stosując odpowiednią obsługę wyjątków. Kod związany z obsługą wyjątków, które mogą wystąpić, należy umieścić w specjalnym bloku `try` ograniczonym nawiasami klamrowymi:

```
try {
// instrukcje
}
```

W bloku `try` należy dokonać tzw. zgłoszenia wyjątku. Odbywa się to za pomocą konstrukcji `throw` wywołującej obsługę wyjątków. Najczęściej stosowanym rozwiązaniem jest dodawanie do konstrukcji argumentu będącego obiektem wbudowanej w PHP klasy `Exception`. Może ona przyjmować (opcjonalnie) dwa argumenty: komunikat błędu oraz kod błędu.

```
throw new Exception("komunikat", kod);
```

Klasa `Exception` ma szereg właściwości i metod umożliwiających przekazywanie informacji na temat danego wyjątku. Metody te zestawiono i opisano w tabeli 26.1.

Tabela 26.1. Metody klasy `Exception`

Metoda	Opis
`getCode()`	Zwraca numeryczny opcjonalny kod wyjątku.
`getFile()`	Zwraca nazwę pliku (pełną ścieżkę), w którym wywołany został wyjątek.
`getLine()`	Zwraca numer wiersza, w którym wywołany został wyjątek.
`getMassage()`	Zwraca opcjonalną treść wyjątku.
`getTrace()`	Zwraca tablicę zawierającą zapis stosu wywołań do miejsca, w którym został zgłoszony wyjątek.
`getTraceAsString()`	Zwraca ślad stosu jako `string`.

Dodatkowo po bloku **try** należy umieścić minimum jeden blok **catch**. Jego zadaniem jest przechwycenie wyjątku, który wystąpił, i wykonanie w związku z tym odpowiednich czynności. Czynności mogą być związane z wyświetleniem informacji dotyczących wyjątku oraz przede wszystkim z podjęciem kroków prowadzących do rozwiązania zaistniałych problemów. Instrukcja **catch** przyjmuje jeden argument złożony z nazwy klasy (**Exception** lub jej klasa podrzędna) i zmiennej, za pomocą której odwołujemy się do występującego wyjątku.

```
catch (nazwa_klasy $zmienna){
//instrukcje
}
```

Przykład z list. 26.1 przedstawia zastosowanie wszystkich bloków potrzebnych do obsłużenia wyjątku. Blok **try** zawiera proste zgłoszenie wyjątku obsługiwane przez konstrukcję **throw** zawierającą wbudowany argument **Exception**, który ma dwa parametry. W bloku **catch** wyprowadzono informacje na temat przechwyconego wyjątku dzięki zastosowaniu odpowiednich metod klasy **Exception**. Efekt działania skryptu prezentuje rys. 26.1.

Listing 26.1

```
<?php
try {
throw new Exception('Uwaga! Wystąpił wyjątek!', 17);
}
catch(Exception $w) {
print ("Wyjątek o kodzie: <b>".$w->getCode()."</b><br>");
print ("Komunikat wyjątku: <b>".$w->getMessage()."</b><br>");
print ("Wyjątek wystąpił w pliku: <b>".$w->getFile().
"</b><br>");
print ("Wyjątek wystąpił w lini: <b>".$w->getLine().
"</b><br>");
}
?>
```

Rys. 26.1. Wynik działania skryptu z list. 26.1 – obsługa wyjątku

Dodatkowo biblioteka SPL dostarcza zbiór gotowych klas wyjątków dla najczęściej pojawiających się problemów. Są to klasy dziedziczące po Exception i wykorzystuje się je analogicznie jak w przykładzie list. 26.1. Powinno się je stosować jak najczęściej w celu uniknięcia przypadków wymienionych w tabeli 26.2.

Tabela 26.2. Wyjątki dla najczęściej pojawiających się problemów

Wyjątek	Opis
RuntimeException	Błędy występujące w trakcie wykonywania (np.: brak pliku, błąd połączenia).
OutOfRangeException	Przekroczenia przez wartość ustalonego zakresu.
RangeException	Podanie niewłaściwego zakresu wartości.
OutOfBoundsException	Brak wartości o podanym kluczu.
DomainException	Użycie niewłaściwych danych.
LengthException	Wartość o niewłaściwej długości.
InvalidArgumentException	Argument o niewłaściwej wartości.
OverflowException	Dodanie nowej wartości do przepełnionego pojemnika na dane.
UnderflowException	Próba pobrania wartości z pustego.
LogicException	Niewłaściwe wyrażenie logiczne.

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Napisz skrypt zawierający funkcję, której zadaniem jest zwrócenie wyniku dzielenia dwóch argumentów tej funkcji. Dodatkowo wprowadź obsługę wyjątku dzielenia przez 0.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Co to jest wyjątek?
2. Przedstaw metody klasy Exception.

27 Obsługa plików

ZAGADNIENIA

- Jakie są funkcje związane z obsługą plików?
- Jakie są tryby otwierania pliku?
- Jak zapisywać dane w pliku za pomocą skryptu PHP?
- Jak wyświetlać na stronie zawartość pliku?

Plik jest podstawową formą przechowywania danych w języku PHP. Dostępny jest szereg funkcji pozwalających na interakcje z plikami zapisanymi na serwerze. Do najważniejszych z nich należy tworzenie, odczytywanie, zapisywanie, usuwanie i przenoszenie plików.

Otwieranie pliku obsługuje w PHP funkcja **`fopen()`** przyjmująca dwa parametry: nazwę pliku i deklarację trybu otwierania (tab. 27.1), który określa, w jaki sposób plik będzie używany. Tryb otwierania może ograniczyć plik tylko do odczytu, określić miejsce dopisania kolejnych danych (początek lub koniec pliku) oraz utworzyć plik, jeżeli nie ma go w bieżącej lokalizacji na serwerze. To, co zostanie zwrócone przez funkcję, najczęściej przypisuje się do zmiennej, co pozwala na łatwiejsze operowanie zawartością pliku.

```
fopen(nazwa_pliku, tryb otwierania);
```

Tabela 27.1. Tryby otwierania plików

Tryb	Opis
`a`	Otwiera plik do dopisywania danych. Dane będą dodawane na końcu pliku.
`a+`	Otwiera plik do odczytu i dopisywania danych. Dane będą dodawane na końcu pliku.
`r`	Otwiera plik tylko do odczytu.
`r+`	Otwiera plik do odczytu i zapisu. Dane będą dodawane na początku pliku.
`w`	Otwiera plik tylko do zapisu. Jeżeli plik istnieje, wszystkie dane zostaną skasowane, a jeżeli nie, PHP spróbuje go utworzyć.
`w+`	Otwiera plik do odczytu i zapisu. Jeżeli plik istnieje, wszystkie dane zostaną skasowane, jeżeli nie, PHP spróbuje go utworzyć.

Zapisywania danych w pliku dokonuje się za pomocą funkcji **`fwrite()`** lub funkcji **`fputs()`**. Obie funkcje standardowo przyjmują dwa parametry (opcjonalnie trzy). Pierwszy to zmienna wskazująca na plik, do którego zostaną zapisane informacje określone

przez parametr drugi (ciąg znaków lub zmienna). Trzeci (opcjonalny) parametr określa maksymalną liczbę bajtów do zapisania.

```
fwrite($zmienna, wprowadzane_dane);
fputs($zmienna, wprowadzane_dane);
```

Kiedy nastąpi zakończenie pracy z plikiem, należy go zamknąć. Wykorzystuje się w tym celu funkcję **fclose()**. Przyjmuje ona jako parametr zmienną wskazującą na plik, który ma zostać zamknięty.

```
fclose($zmienna);
```

Funkcja ta zwraca wartość **true**, jeżeli plik został zamknięty, lub **false**, jeżeli zamknięcie pliku się nie powiodło.

W wypadku gdy chcemy odczytać zawartość pliku, w pierwszej kolejności należy go otworzyć. Następnie trzeba zastosować pętlę **while**, zawierającą warunek sprawdzający, czy został osiągnięty koniec pliku (**eof** – z ang. *end of file*). Wewnątrz pętli należy umieścić funkcję **fgets()**. Jej zadaniem jest odczytanie zawartości pliku wiersz po wierszu. Przyjmuje ona dwa parametry: zmienną wskazującą na plik oraz wartość określającą liczbę bajtów pliku do przeczytania pomniejszoną o jeden.

```
while(!feof($zmienna)){
fgets($zmienna,wartość);
}
```

Po zakończonej pracy należy pamiętać o zamknięciu pliku funkcją **fclose()**.

Możliwe jest przeczytanie pliku w jednym przejściu bez rozdrabniania się na pojedyncze wiersze. Pozwala na to funkcja **freadfile()** przyjmująca jako parametr nazwę pliku lub całą ścieżkę dostępu do pliku.

```
freadfile(nazwa_pliku);
```

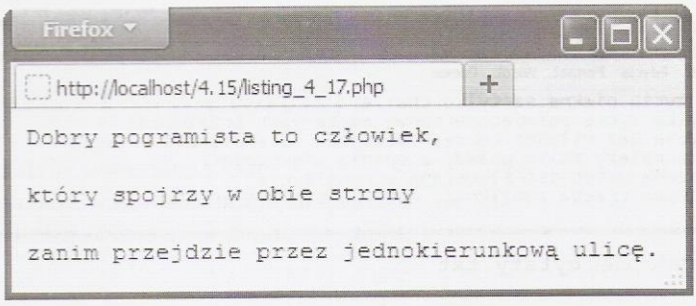
Wywołanie powyższej funkcji powoduje otwarcie pliku, wyświetlenie jego zawartości w przeglądarce oraz zamknięcie pliku.

Przykład z list. 27.1 prezentuje podstawowe funkcje związanie z obsługą pliku. W pierwszej kolejności plik **plik.txt** zostaje utworzony, jeżeli nie istnieje (tryb otwierania **w+**), i otwarty za pomocą funkcji **fopen()**. Plik przypisano do zmiennej **$plik**, poprzez którą można w łatwy sposób odwołać się do pliku. Następnie zapisano w pliku sentencję składającą się z trzech linii tekstu, ustawioną jako tekst preformatowany. Po zakończeniu operacji zapisywania plik został zamknięty. Ponowne otwarcie pliku nastąpiło z trybem otwarcia **r**. Pętla **while** sprawdza, czy do zmiennej **$wiersz** przypisano pojedynczą linijkę tekstu z pliku **plik.txt**. Jeżeli warunek jest spełniony, zawartość zmiennej **$wiersz** zostaje wyświetlona w przeglądarce. Tym sposobem na stronie (rys. 27.1) pojawia się cała zawartość pliku tekstowego.

Listing 27.1

```
<?php
$plik = fopen("plik.txt","w+");
fputs($plik,"<pre>Dobry pogramista to człowiek,\n
który spojrzy w obie strony \n
zanim przejdzie przez jednokierunkową ulicę.</pre>");
fclose($plik);
```

```
$plik = fopen("plik.txt","r");
while($wiersz = @fgets($plik, 1024)){
print ($wiersz);
}
fclose($plik);
?>
```



Rys. 27.1. Wynik działania skryptu z list. 27.1 – obsługa pliku

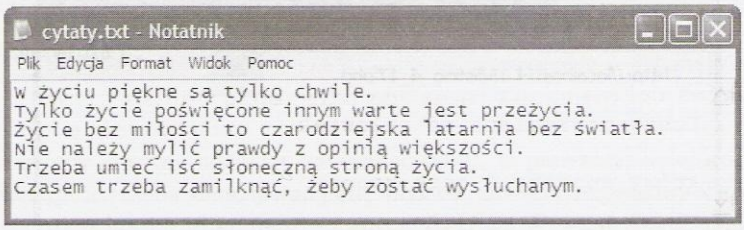
Język PHP udostępnia szereg funkcji pozwalających na wyświetlenie informacji na temat danego pliku. Funkcje te jako parametr przyjmują zmienną, do której przypisano konkretny plik. Tabela 27.2 zawiera zestaw funkcji określających konkretne parametry pliku.

Tabela 27.2. Funkcje informacyjne

Tryb	Opis
`fileatime()`	Czas ostatniego odczytu pliku.
`filectime()`	Czas ostatniej modyfikacji i-węzła (dotyczy tylko systemów Unix) w formacie timestamp.
`filemtime()`	Czas ostatniej modyfikacji pliku.
`fileowner()`	Identyfikator użytkownika, który jest właścicielem pliku.
`filegroup()`	Identyfikator grupy, do której należy plik (tylko Unix).
`fileinode()`	Numer i-węzła, do którego przypisany jest plik (tylko Unix).
`fileperms()`	Prawa dostępu do pliku.
`filesize()`	Wielkość pliku w bajtach.
`filetype()`	Typ pliku (tylko Unix).
`stat()`	Tablica zawierająca pełne informacje o pliku.

Plik jako podstawowa jednostka przechowywania danych może być wykorzystywany w wielu skryptach PHP – począwszy od liczników odwiedzin, a skończywszy na zapisie danych z formularzy (np. księga gości).

Przykład z list. 27.2 przedstawia wykorzystanie pliku do wyświetlania na stronie losowych cytatów. Plik **cytaty.txt** (rys. 27.2) zawiera sześć sentencji. Każda z nich zapisana jest w nowej linii.

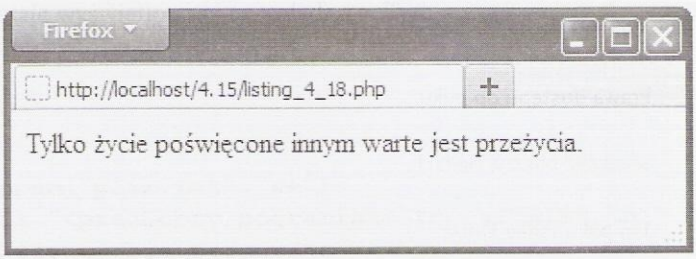


Rys. 27.2. Zawartość pliku **cytaty.txt**

Po otwarciu pliku skrypt wprowadza pojedynczo sentencje do tablicy (**$tablica**) do czasu napotkania końca pliku. Następnie plik zostaje zamknięty. Do zmiennej **$i** przypisano funkcję losującą wartość liczbową z przedziału od 0 do 5. W zależności od wylosowanej wartości, która stanowi indeks tablicy, na stronie (rys. 27.3) pojawi się sentencja przypisana do wylosowanego indeksu **$i**. Liczba sentencji może zostać zwiększona. Należy wówczas pamiętać o wprowadzeniu korekty w zakresie parametrów funkcji **rand()**.

Listing 27.2

```
<?php
$tablica = array();
$i = 0;
$plik = fopen("cytaty.txt","r");
while(!feof($plik)){
$tablica[$i]=fgets($plik, 1024);
$i++;
}
fclose($plik);
$i=rand(0,5);
print ($tablica[$i]);
?>
```



Rys. 27.3. Wynik działania skryptu z list. 27.3 – losowa sentencja

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Napisz skrypt, który wpisze do pliku liczby parzyste i podzielne przez 3 z przedziału od 1 do 1000, a następnie odczyta zawartość pliku i wyświetli go w oknie przeglądarki.
2. Napisz skrypt odczytujący cztery liczby zapisane w pliku tekstowym (każda w osobnej linii) i wypisujący je w oknie przeglądarki w kolejności malejącej.
3. Napisz skrypt zapisujący do pliku 10 losowych liczb z przedziału od 0 do 100. Odczytaj z pliku największą z wylosowanych liczb.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Jakie znasz tryby otwierania pliku?
2. Jaka funkcja odpowiada za zamknięcie pliku?
3. Podaj trzy funkcje określające parametry pliku.

28 Obsługa formularzy

ZAGADNIENIA

- Jakie właściwości ma znacznik <form>?
- Jak odwołać się do pojedynczego elementu w formularzu?
- Jak przesyłać dane zawarte w formularzu na inną stronę?
- Jak pobierać i przetwarzać dane wprowadzone do formularza?

Formularze w skrypcie PHP pełnią kluczową funkcję. Pomagają gromadzić i przetwarzać dane wprowadzone przez użytkownika. Stanowią podstawę budowy różnych elementów strony, takich jak: księga gości, sondy, ankiety, formularze zamówieniowe w sklepach i wiele innych.

Sam formularz zbudowany jest ze znaczników języka HTML. W przypadku powiązania go ze skryptem ważne jest określenie właściwości znacznika **<form>**. Ogólny zapis formularza przedstawia się następująco:

```
<form name = "nazwa"
   target = "okno"
   action = "url"
   method = "metoda"
   enctype = "typ kodowania">
<!-- definicja obiektów składowych-->
</form>
```

Właściwości, które są najistotniejsze, to **action** i **method**. **Action** określa adres skryptu, który będzie odbierał dane wprowadzone do formularza. Parametr **method** wskazuje metodę, która zostanie zastosowana do przesłania danych do serwera. W tym przypadku dostępne są dwie możliwości:

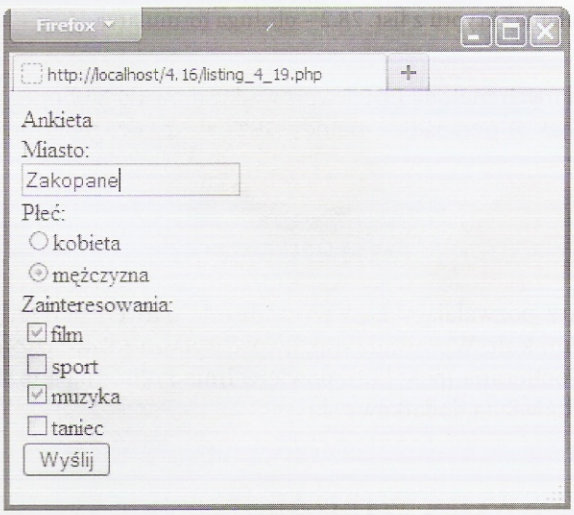
- **GET** – przekazywane dane zostają dołączone do adresu URL,
- **POST** – poufne przekazanie danych.

Metoda **GET** stosowana jest zazwyczaj w prostych formularzach, gdzie nie ma konieczności ukrywania przekazywanych danych. W wypadku rozbudowanych formularzy wykorzystuje się metodę **POST**.

Przykład z list. 28.1 prezentuje prosty formularz (rys. 28.1) zapisany w języku HTML. Zawiera on trzy rodzaje pól: pole tekstowe (**text**), pole opcji (**radio**) oraz pole wyboru (**checkbox**). Po wciśnięciu przycisku zatwierdzającego **Wyślij** następuje przejście do skryptu **listing_4_20.php**. Dane wprowadzone do formularza zostają jawnie przekazane do skryptu za pomocą metody **GET**.

Listing 28.1

```
<form action = "listing_4_20.php" method = "GET">
Ankieta<br>
Miasto:<br>
<input type="text" name="miasto"><br>
Płeć:<br>
<input type="radio" name="plec" value="k">kobieta<br>
<input type="radio" name="plec" value="m">mężczyzna<br>
Zainteresowania:<br>
<input type="checkbox" name="hobby[]" value="film">film<br>
<input type="checkbox" name="hobby[]" value="sport">sport<br>
<input type="checkbox" name="hobby[]" value="muzyka">muzyka<br>
<input type="checkbox" name="hobby[]" value="taniec">taniec<br>
<input type="submit" value="Wyślij">
</form>
```



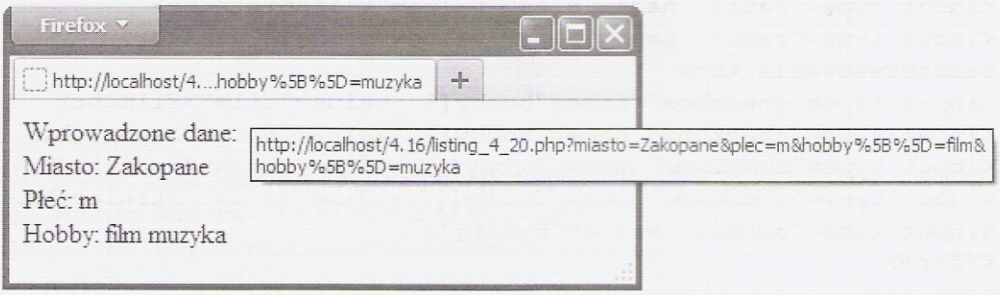
Rys. 28.1. Wynik działania skryptu z list. 28.1 – formularz

Listing 28.2 przedstawia skrypt, którego zadaniem jest wyświetlanie danych wprowadzonych do formularza. Dane te przechowywane są w tablicy **$_GET**, indeks tablicy w tym wypadku stanowi nazwa (**name**) odnosząca się do odpowiedniego pola w formularzu. W przypadku pól wyboru (**checkbox**) istnieje możliwość wybrania więcej niż jednej odpowiedzi. Dlatego przed wypisaniem odpowiedzi skrypt sprawdza (**!empty ($_GET['hobby'][$i])**), czy do tablicy została wprowadzona jakaś wartość (czy zostało zaznaczone pole w formularzu).

Listing 28.2

```
<?php
print("Wprowadzone dane:<br>");
print("Miasto: ".$_GET['miasto']."<br>");
print("Płeć: ".$_GET[,plec']."<br>");
```

```
print("Hobby: ");
for($i=0;$i<4;$i++) {
if(!empty($_GET['hobby'][$i])) {
print($_GET['hobby'][$i]." ");
} }
?>
```



Rys. 28.2. Wynik działania skryptu z list. 28.2 – obsługa formularza

Przekazywanie danych metodą **POST** wprowadza dane do tablicy **$_POST**. Zasada działania jest podobna jak w przypadku metody **GET**, ale z tą różnicą, że wartości nie są widoczne w polu adresu.

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Zbuduj formularz pozwalający na wprowadzenie danych potrzebnych do obliczenia pola i powierzchni kuli. Wprowadź możliwość wyboru (**checkbox**), która z tych dwu wartości będzie obliczana (pole kuli, powierzchnia kuli), mogą też być obliczane obie. Za obliczenia odpowiada dodatkowy skrypt.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Jakie właściwości przyjmuje znacznik **<form>**?
2. W jaki sposób pobieramy dane wprowadzone do formularza?

29 Współpraca PHP i MySQL

ZAGADNIENIA

- Co to jest relacyjna baza danych?
- Jak PHP współpracuje z bazą danych?
- Jak łączyć się z bazą danych za pomocą języka PHP?
- Jak wysyłać zapytania do bazy?

Wygodniejszym i wydajniejszym sposobem przechowywania i korzystania z danych zapisanych na serwerze jest współpraca z relacyjną bazą danych. Baza taka składa się z tabel powiązanych ze sobą relacjami, które pozwalają na logiczne łączenie danych z różnych tabel.

Połączenie z bazą danych w języku PHP odbywa się za pomocą funkcji **mysql_connect()**. Przyjmuje ona trzy argumenty: adres serwera (adres IP lub nazwa hosta), nazwę użytkownika i hasło. W przypadku prawidłowego połączenia funkcja zwraca identyfikator połączenia. Jeżeli połączenie nie zostanie nawiązane, funkcja zwraca wartość **false**.

```
$polaczenie = mysql_connect(adres_serwera, nazwa_użytkownika,
hasło);
```

Kolejnym krokiem jest wybranie bazy danych, na której będą wykonywane operacje. Umieszczenie znaku „małpa" @ przed funkcją zapobiega wyświetleniu komunikatu o błędzie w wypadku, gdy baza danych nie zostanie odnaleziona.

```
$db = @mysql_select_db('nazwa_bazy', $polaczenie);
```

Do wybranej bazy danych można przesyłać zapytania związane z tworzeniem, modyfikacją i usuwaniem tabel, pobieraniem danych zawartych w tabelach oraz ich modyfikacją. Funkcją związaną z wykonaniem zapytania do bazy jest **mysql_query()**. Funkcja jako argument przyjmuje zmienną wskazującą na zapytanie napisane w języku obsługującym bazę. W przypadku MySQL może to być na przykład proste zapytanie zwracające zawartość wszystkich wierszy w wybranej tabeli, które przypisujemy do zmiennej:

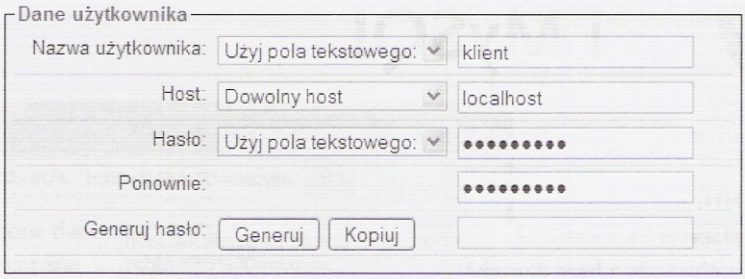
```
$zapytanie="select*from nazwa_tabeli";
```

Po wykonaniu pracy w bazie danych należy pamiętać o zamknięciu połączenia:

```
mysql_close($polaczenie);
```

Każda baza danych powinna mieć dodatkowego użytkownika, któremu można nadać odpowiednie uprawnienia działań na bazie. Można go utworzyć w panelu administracyjnym phpMyAdmin. Jest on również dostępny w oprogramowaniu EasyPHP pod adresem http://127.0.0.1/home/mysql/index.php. Następnie w zakładce **Uprawnienia** należy wybrać opcję **Dodaj nowego użytkownika**. W oknie przeglądarki pojawi się nowy panel (rys. 29.1), w którym wprowadzamy takie informacje jak nazwa użytkownika (klient), host

(localhost), hasło (fanfilmow) oraz odpowiednie uprawnienia. Uprawnienia pozwalają na określenie możliwości użytkowania w trzech obszarach: danych, struktury oraz administracji. Po zaznaczeniu odpowiednich opcji wystarczy już tylko wybrać przycisk **Wykonaj**.



Rys. 29.1. Tworzenie nowego użytkownika w phpMyAdmin

W celu połączenia się z bazą danych należy stworzyć bazę, utworzyć w niej tabele oraz wprowadzić dane. PhpMyAdmin udostępnia zakładkę SQL, w której istnieje możliwość wprowadzenia ciągu instrukcji w języku MySQL (list. 29.1), pozwalających za jednym razem utworzyć prostą strukturę bazy.

Listing 29.1

```
CREATE DATABASE baza_filmow;
CREATE TABLE filmy (
ID INT NOT NULL AUTO_INCREMENT, PRIMARY KEY (ID),
tytul VARCHAR(35),
rezyser VARCHAR(35),
czas INT);
INSERT INTO filmy VALUES (NULL, "Pan Tadeusz", "A.Wajda", 207);
INSERT INTO filmy VALUES (NULL, "Matrix", "A.Wachowski", 196);
INSERT INTO filmy VALUES (NULL, "Shrek", "A.Adamson", 150);
```

W wyniku przeprowadzonych operacji (list. 29.2) została utworzona baza danych o nazwie **baza_filmow**. Zawiera ona jedną tabelę **filmy** (rys. 29.2), która ma 4 pola: **id** – klucz główny mający zdolność autonumeracji, **tytul**, **reżyser** oraz **czas**. Do tabeli wprowadzono trzy rekordy danych.

	ID	tytul	rezyser	czas
☐ ✎ ✕	1	Pan Tadeusz	A.Wajda	207
☐ ✎ ✕	2	Matrix	A.Wachowski	196
☐ ✎ ✕	3	Shrek	A.Adamson	150

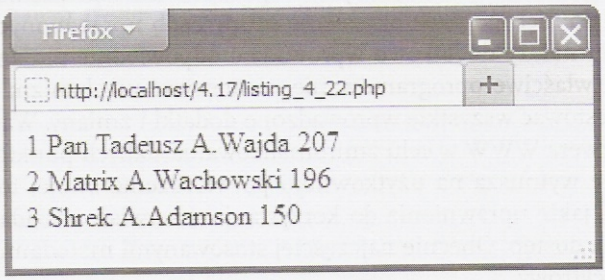
Rys. 29.2. Widok tabeli „filmy" w phpMyAdmin

Mając bazę danych wypełnioną danymi, można przystąpić do wykonania połączenia z poziomu języka PHP. W przypadku instrukcji połączenia z bazą, wyboru bazy i wykonania zapytania warto wprowadzić dodatkową konstrukcję **or die()**, która w przypadku niepowodzenia pierwszej instrukcji wyprowadzi w oknie strony informację tekstową i przerwie działanie skryptu.

Listing 29.2

```
<?php
$polaczenie=mysql_connect("localhost","klient","fanfilmow")
or die("Brak połączenia z serwerem MySQL");
mysql_select_db("baza_filmow",$polaczenie)
or die("Błąd wyboru bazy danych");
$zapytanie="SELECT * FROM filmy";
$wynik=mysql_query($zapytanie)
or die ("Wystąpiły problemy przy zapisywaniu danych");
while ($wiersz_danych = mysql_fetch_row($wynik)) {
for ($i=0;$i<count($wiersz_danych);$i++)
print $wiersz_danych[$i]." ";
print "<Br>";
}
mysql_close($polaczenie);
?>
```

Skrypt (list. 29.2) w pierwszej kolejności nawiązuje połączenie z bazą danych. Następnie dokonany zostaje wybór bazy. Kolejnym krokiem jest wprowadzenie zapytania do baz – w tym przypadku mają zostać wybrane wszystkie rekordy zawarte w tabeli **filmy**. Wynik zapytania zostaje wyświetlony w oknie przeglądarki (rys. 29.3) za pomocą pętli **while**. Ostatnim etapem jest zamknięcie połączenia z bazą **baza_filmow**.



Rys. 29.3. Wynik działania skryptu z list. 29.2 – połączenie z **baza_filmow**

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Zmodyfikuj skrypt z listingu 29.2, rozszerzając go o dodatkowe zapytanie wprowadzające nowy rekord do tabeli **filmy**.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Jaka funkcja odpowiada za połączenie z bazą danych?
2. Jakie zapytania obsługuje baza danych MySQL?

30 Zabezpieczanie witryn WWW

ZAGADNIENIA

- Jakie zagrożenia dotyczą bezpieczeństwa strony WWW?
- Jakie znaczenie ma uwierzytelnianie?
- Jak zabezpieczyć dane na stronie?

Podczas tworzenia zarówno własnych, jak i komercyjnych stron internetowych należy zwrócić szczególną uwagę na bezpieczeństwo. Zagrożenia, które mają wpływ na bezpieczeństwo strony, mogą być związane z ujawnieniem informacji poufnych, utratą danych, modyfikacją danych i zablokowaniem usług. Jest to tylko kilka podstawowych obszarów, których bezpieczeństwo ma wpływ na prawidłowe funkcjonowanie strony internetowej.

Pewna część informacji przechowywanych w serwisie ma charakter poufny. Mogą to być na przykład dane osobowe, hasła czy dokumenty firmowe. Ujawnienie takich danych może mieć bardzo poważne skutki. Jednym ze sposobów zabezpieczenia danych jest ograniczenie dostępu do nich poprzez określenie grupy osób uprawnionych do ich przeglądania. W tym celu należy dodatkowo wprowadzić odpowiednie zmiany w parametrach serwera, dołączyć właściwe oprogramowanie, zastosować mechanizm uwierzytelniania, a następnie przetestować wszystkie wprowadzone dodatki i zmiany. Warto usunąć niepotrzebne usługi serwera WWW w celu zminimalizowania słabych punktów systemu.

Uwierzytelnianie wymusza na użytkowniku potwierdzenie swojej tożsamości. Dzięki temu system wie, jakie uprawnienia do korzystania z serwisu posiada użytkownik i do jakich zasobów ma dostęp. Obecnie najczęściej stosowanymi metodami uwierzytelniania są: hasło i podpis cyfrowy.

Ujawnienie danych może nastąpić również w trakcie ich przesyłania w sieci. W celu ich odpowiedniego zabezpieczenia należy je zaszyfrować przed wysłaniem i odszyfrować w miejscu docelowym. Stosuje się do tego protokół SSL opracowany przez firmę Netscape.

Kolejnym zagrożeniem jest możliwość utraty danych. Może być ona następstwem celowego ataku lub nieszczęśliwego wypadku (przypadkowe skasowanie danych lub uszkodzenie sprzętu). Najlepszą metodą zabezpieczenia się jest częste i systematyczne tworzenie kopii zapasowych. Warto też dobrać w tym celu odpowiednie narzędzia, ułatwiające cały proces, oraz zapisywać kopie na osobnych nośnikach danych.

Czasem jednym z najtrudniejszych do wykrycia zagrożeń jest modyfikacja danych na stronie. Zdarza się, że jest to bezpośrednia i widoczna zmiana mająca na celu pozostawienie na stronie znaku swojej obecności. Niejednokrotnie jednak są to pojedyncze zmiany w plikach lub danych znajdujących się na serwerze, które nie są widoczne na pierwszy rzut oka, a mogą mieć poważne konsekwencje. W celu zabezpieczenia się przed atakami powinno się wykonywać obliczanie sumy kontrolnej plików przesyłanych na serwer. Suma kontrolna jest to wartość liczbowa wyliczana dla wyznaczonej porcji danych, zależna od zawartości danego pliku. Dowolna nawet najmniejsza modyfikacja pliku spowoduje zmianę

wartości sumy kontrolnej. Dodatkowym zabezpieczeniem jest zaszyfrowanie danych, co utrudnia ich bezpośrednią modyfikację.

Zagrożeniem, na które narażona jest każda strona, jest zablokowanie usługi. Może to przybrać postać całkowitego zablokowania dostępu do usługi lub zaistnienia znacznych opóźnień czasowych. Jest to najczęściej związane z masowym wysyłaniem zapytań do serwera, który w pewnym momencie nie może poradzić sobie z ich obsługą. Inny sposób to umieszczenie na serwerze złośliwego oprogramowania, które spowolni jego pracę. Ochrona przed takimi atakami jest złożona. Trzeba monitorować, które porty najczęściej wykorzystywane są podczas ataków, i w odpowiedniej chwili je zablokować. Należy też analizować żądania pod względem wykorzystywanych protokołów.

Dostępne na serwerze oprogramowanie, jak również własne programy, mogą zawierać luki pozwalające na łatwy dostęp do danych. Dlatego bardzo ważnym zadaniem jest dogłębne przetestowanie serwisu pod względem funkcjonowania i bezpieczeństwa.

Do dotychczas omówionych zagrożeń należy dodać zagrożenia fizyczne. System, na którym znajduje się strona internetowa, powinien być umieszczony w osobnym, zamkniętym pomieszczeniu. Dostęp do niego powinny mieć jedynie wyznaczone osoby. Pomieszczenie powinno być klimatyzowane, a sprzęt – posiadać zabezpieczenia związane z utratą zasilana czy uszkodzeniem sieci.

SPRAWDŹ SWOJĄ WIEDZĘ

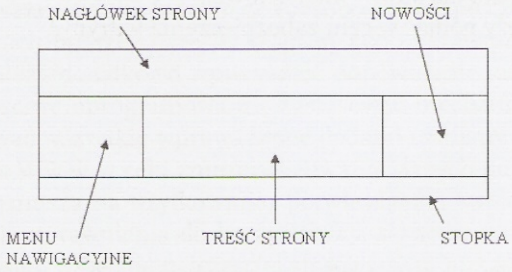
1. Jakie znasz zagrożenia dla strony internetowej?
2. Jakie działania należy podjąć w celu zabezpieczenia witryny?

31 Praktyczne przykłady

Gdy zna się elementy takich języków jak HTML, CSS, JavaScrip i PHP, można rozpocząć budowanie złożonych stron internetowych wzbogaconych o dodatkowe aplikacje i bazy danych. Pamiętaj o walorach estetycznych tworzonych projektów. W odpowiedni sposób dobieraj tło, kolory i krój czcionek. Dostosuj kolorystykę linków zwykłych, odwiedzonych i aktywnych lub wprowadź ich odpowiedniki graficzne. W przypadku własnych projektów pamiętaj o zachowaniu praw autorskich, jeżeli wykorzystasz elementy (teksty, grafikę) autorstwa innych osób.

PRZYKŁAD 1

Utwórz stronę internetową (sklep z zaproszeniami) o zaplanowanych poniżej elementach.



Rys. 31.1. Układ strony WWW dla przykładu 1

- W menu nawigacyjnym umieść trzy odnośniki (linki) do podstron: strona główna, zaproszenia, kontakt. W nagłówku umieść samodzielnie wykonane logo sklepu lub banner (Indywidualne zaproszenia). W części nowości wyświetl za pomocą skryptu jedno z pięciu losowych zaproszeń. Grafika ulega zmianie przy każdym odświeżeniu strony. W stopce wprowadź dane autora (imię i nazwisko), datę utworzenia i nazwę oprogramowania wykorzystanego do stworzenia witryny.
- Na stronie głównej zamieść opis sklepu:

  Indywidualne zaproszenia

  Każda uroczystość zaczyna się od zaproszenia...

  Nasza strona internetowa umożliwia wybranie własnych wymarzonych zaproszeń, które swoim oryginalnym wyglądem i wysokiej jakości wykonaniem z pewnością mile zaskoczą Państwa gości.

  Dajemy Państwu możliwość indywidualnego wyboru projektu zaproszenia oraz dopasowywania kolorystyki, zmiany tekstów i dodawania zdjęć.

Tu stworzą Państwo indywidualne zaproszenia!
Oferujemy:
- Szeroki wybór szablonów
- Najwyższą jakoś druku
- Szybki termin realizacji

- Podstrona „Zaproszenia" zawiera galerię prezentującą obrazy zaproszeń (min. 9). Galeria przedstawia miniatury obrazów o jednakowych rozmiarach. Po kliknięciu wybranego zaproszenia w nowym oknie pojawia się obraz w powiększeniu.
- Podstrona „Kontakt":

Indywidualne zaproszenia
ul. Starowiejska 120

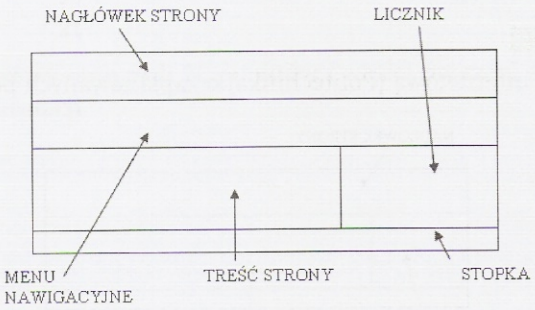
tel. (55) 632-59-36 biuro
tel kom. 566-172-777

czynne: pon.–pt. 9^{00}–18^{00}, sob. 10^{00}–14^{00}

Zapytania i zamówienia próbek zaproszeń prosimy wysyłać na poniższy adres:
izaproszenia@zaproszenia.pl

PRZYKŁAD 2

Utwórz stronę internetową (książka kucharska) o zaplanowanych poniżej elementach.



Rys. 31.2. Układ strony WWW dla przykładu 2

- W menu nawigacyjnym umieść trzy odnośniki (linki) do podstron: strona główna, przepisy, przelicznik wag. W nagłówku umieść samodzielnie wykonany banner (Książka kucharska). W części „Licznik" wyświetl samodzielnie wykonany licznik odwiedzin (skrypt) oraz aktualną datę (skrypt) w następującej formie, np. piątek, 22 listopada 2012. W stopce wprowadź dane autora (imię i nazwisko), datę utworzenia i nazwę oprogramowania wykorzystanego do stworzenia witryny.
- Na stronie głównej zamieść opis strony:
  Książka kucharska
  Na stronie znajdziesz przepisy i porady kulinarne. Jeżeli chcesz, podziel się z nami swoim przepisem. Prześlij go na adres: przepisy@książkakucharska.pl. Każdy z przepisów uporządkuj w następujący sposób:

- Składniki
- Wykonanie
- Czas pracy
- Zdjęcie

Bo przepisy są po to, by się nimi podzielić.

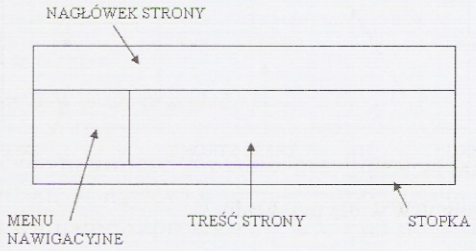
- Podstrona „Przepisy" zawiera przykładowe przepisy na 5 dań. Do każdego z przepisów dodaj zdjęcie.
- Podstrona „Przelicznik" wag zawiera formularz pozwalający na przeliczenie wagi kilku produktów podanych poniżej. Przelicznik powinien uwzględniać takie elementy jak: rodzaj produktu, przeliczaną wagę, z jakiej jednostki (np. dag, łyżka), na jaką jednostkę (szklanka). Do formularza zostaje wprowadzona nazwa i waga produktu, skrypt powoduje przeliczenie na jednostki typu łyżka, szklanka i wyświela wynik, jak w przykładowym rozwiązaniu.

produkt	łyżka	szklanka
cukier	1,5 dag	25 dag
kakao	0,6 dag	10 dag
mąka pszenna	0,9 dag	15 dag
mleko	1,5 dag	25 dag

Przykładowe rozwiązanie: cukier: 25 dag = 1 szklanka.

PRZYKŁAD 3

Utwórz stronę internetową (Politechnika) o zaplanowanych poniżej elementach.



Rys. 31.3. Układ strony WWW dla przykładu 3

- W menu nawigacyjnym umieść trzy odnośniki (linki) do podstron: strona główna, kierunki, wyniki. W nagłówku umieść samodzielnie wykonany banner (Politechnika), banner zmienia się po najechaniu na niego myszką na grafikę zawierającą adres politechniki.

Politechnika
al. Jerozolimskie 12
35-959 Rzeszów

W stopce wprowadź dane autora (imię i nazwisko), datę utworzenia i nazwę oprogramowania wykorzystanego do stworzenia witryny.

- Na stronie głównej zamieść opis:
  Politechnika kształci i podejmuje zadania naukowo-badawcze w dziedzinie nauk technicznych, matematycznych, fizycznych, chemicznych, ekonomicznych i biologicznych.
  Poprzez współpracę z przemysłem doskonali programy kształcenia przygotowujące absolwentów do aktywnego uczestniczenia w życiu społecznym i gospodarczym, zarówno w wymiarze lokalnym, jak i narodowym.
  Nasze atuty:
  - Doskonała lokalizacja
  - Nowoczesne laboratoria
  - Znakomita kadra
- Podstrona „Kierunki" prezentuje opis 3 wiodących kierunków politechniki: informatyka, mechatronika i budownictwo.
- Podstrona „Wyniki" zawiera prosty formularz logowania (pole tekstowe – login, pole tekstowe – hasło oraz przycisk – zaloguj). Po zalogowaniu (`login: kandydat3`, `hasło:student3`) pojawia się strona wyświetlająca listę kandydatów oraz ich punkty, dane pobierane są z bazy danych MySQL.
  Strona po zalogowaniu:
  Wyniki:

Kandydat	Punkty
Kandydat1	12
Kandydat2	23
Kandydat3	20
Kandydat4	15
Kandydat5	21

  Kandydaci, którzy otrzymali 20 punktów i więcej proszeni są o osobisty kontakt z działem rekrutacji.

WYKAZ PODSTAWOWYCH POJĘĆ W JĘZYKACH POLSKIM, ANGIELSKIM I NIEMIECKIM

JĘZYK POLSKI	JĘZYK ANGIELSKI	JĘZYK NIEMIECKI
blok	block	der Block
dekrementacja	decrement	das Dekrementieren
destruktor	destructor	der Destruktor
domena	domain	die Domäne
dziedziczenie	inheritance	die Erbschaft/ die Vererbung
formularz	form	das Formular
funkcja	function	die Funktion
hosting	hosting	das Hosting
inkrementacja	increment	die Inkrementation/ die Erhöhung
instrukcja przetwarzania warunkowego	conditional processing statement	die Verarbeitungsanweisung
instrukcja warunkowa	conditional statement	die Bedingungsanweisung
instrukcja wyboru	switch statement	die Anwahl-Anweisung
interpreter języka	language interpreter	der Interpreter
klasa	class	die Klasse
kodowanie	coding	die Kodierung
konkatencja	concatenation	die Verkettung
konstruktor	constructor	der Konstruktor
limit transferu	transfer limit	die Transfergrenze/ die Übermittlungsgrenze
lista	list	die Liste
lista rozwijana	drop-down list box	die Auswahlliste/ das Listenfeld
metainformacje	meta-information	die Metadaten
nagłówek dokumentu	document heading/ header	die Hauptüberschrift
obiekt	object	das Objekt
obszar tekstowy	text box	der Textbereich/ das Textfeld
odsyłacz	reference	der Hyperlink
okno decyzyjne	decisive box	das Entscheidungsfenster
okno dialogowe	dialog box	das Dialogfeld
okno tekstowe	text box	das Textfeld
operator kontroli błędów	error control operator	der Fehlerkontrolle-Operator

JĘZYK POLSKI	JĘZYK ANGIELSKI	JĘZYK NIEMIECKI
operator łańcuchowy	chain operator	der Kettenoperator
operator przypisania	assignment operator	der Zuordnungsoperator
operatory arytmetyczne	arithmetic operator	der Rechenoperator/ der Mathematischer Operator
operatory bitowe	bitwise operator	der Bitoperator
operatory logiczne	logical operator/ logical connective	der Verknüpfungsoperator
operatory porównania	comparison operator	der Vergleichsoperator
optymalizacja	optimization	die Optimierung
pętla	loop	die Schleife
portal	portal	der Portal
pozycjonowanie	positioning/ SEO - Search Engine Optimization	die Suchergebnisliste/ das Positionieren
roboty	robots	die Roboter
rozdzielczość	resolution	die Auflösung
serwer	server	der Server
stała	constant	die Konstante
strona internetowa	website/ web page	die Internetseite/ die Webseite
tablica	board/ array	die Tafel
typ liczbowy	numeric type	der Numerischer Typ
typ logiczny	logical type	der Logischer Typ
typ łańcuchowy	string type	der Kettentyp
typ obiektowy	object type	der Objekt Typ
typ złożony	structured type	der Komplexer Typ
typy specjalne	special types	der Spezieller Typ
tytuł	title	der Titel
witryna	web site	die Internetseite
wyjątek	exception	die Ausnahme
zdarzenia	occurrences/ events	das Ereignis
zmienna	variable	die Variable
znacznik	marker	die Markierung/ der Marker/ der Tag
znaki diakrytyczne	diacritic	die Diaktrische Zeichen
znaki specjalne	special characters	die Sonderzeichen/ die speziale Zeichen