

# Praca z serwerem WWW i podstawy PHP

Na podstawie 17 i 19 rozdziału podręcznika  
Internet & World Wide Web  
How to Program, 5/e

Copyright © Pearson, Inc. 2013. All Rights Reserved.

- 17.1 Wstęp
- 17.2 Transakcje HTTP
- 17.3 Architektura aplikacji wielowarstwowych
- 17.4 Programowanie po stronie klienta a programowanie po stronie serwera
- 17.5 Uzyskiwanie dostępu do serwerów WWW
- 17.6 Instalacja Apache, MySQL i PHP
  - 17.6.2 Uruchamianie XAMPP
  - 17.6.3 Testowanie konfiguracji
- 19.1 Wstęp
- 19.2 Prosty program PHP
- 19.3 Konwersja typów
- 19.4 Operatory arytmetyczne
- 19.5 Tablice PHP
- 19.6 Porównywanie łańcuchów
- 19.7 Przetwarzanie łańcuchów i wyrażenia regularne
- 19.8 Przetwarzanie formularzy i logika biznesowa

©1992-2012 by Pearson Education, Inc.  
All Rights Reserved.

## 17.1 Wstęp

- ▶ Serwer WWW odpowiada na żądania klienta (zazwyczaj z przeglądarki internetowej) poprzez dostarczanie zasobów takich jak dokumenty HTML.
- ▶ Gdy użytkownik wprowadzi do przeglądarki adres URL (Uniform Resource Locator), np. [www.deitel.com](http://www.deitel.com), żądany jest określony dokument z serwera WWW.
- ▶ Serwer WWW mapuje URL do zasobu na serwerze i zwraca żądany zasób klientowi.
- ▶ Serwer WWW i klient komunikują się, używając niezależnego od platformy protokołu HTTP (HyperText Transfer Protocol), służącego do transferu żądań i plików przez Internet lub intranet.

©1992-2012 by Pearson Education,  
Inc. All Rights Reserved.

## 17.2 Transakcje HTTP

- ▶ URI (Uniform Resource Identifiers) identyfikuje zasób w Internecie.
- ▶ URI zaczynające się od `http://` są nazywane URL (Uniform Resource Locators).
- ▶ Zazwyczaj URL odnoszą się do plików, katalogów lub kodu po stronie serwera, który wykonuje zadania, takie jak odpytywanie bazy danych, wyszukiwanie zasobów Internetu, czy przetwarzanie aplikacji biznesowych.
- ▶ Innymi słowy, URL zawiera informacje, które kierują przeglądarkę do zasobu, do którego użytkownik chce mieć dostęp.
- ▶ `http://` wskazuje, że do uzyskania zasobu powinien zostać użyty protokół HTTP (HyperText Transfer Protocol).

©1992-2012 by Pearson Education,  
Inc. All Rights Reserved.

## 17.2 Transakcje HTTP (cd.)

- ▶ Następną częścią URL jest pełna nazwa hosta
  - nazwa komputera służącego za serwer WWW, na którym jest zasób.
- ▶ Nazwa hosta jest tłumaczona na adres IP (Internet Protocol) – wartość liczbową, która jednoznacznie identyfikuje serwer.
  - Serwer DNS (Domain Name System) utrzymuje bazę danych nazw hostów z przydzielonymi im adresami IP i automatycznie wykonuje tłumaczenie.
- ▶ Pozostała część URL po nazwie hosta określa lokalizację zasobu na serwerze WWW (np. `/books`) i jego nazwę (np. `downloads.html`).
- ▶ Ze względu na bezpieczeństwo lokalizacja jest zazwyczaj wirtualnym katalogiem.
- ▶ Serwer WWW tłumaczy wirtualny katalog na prawdziwą lokalizację na serwerze, ukrywając w ten sposób przed klientem prawdziwą lokalizację zasobu.

©1992-2012 by Pearson Education,  
Inc. All Rights Reserved.

## 17.2 Transakcje HTTP (cd.)

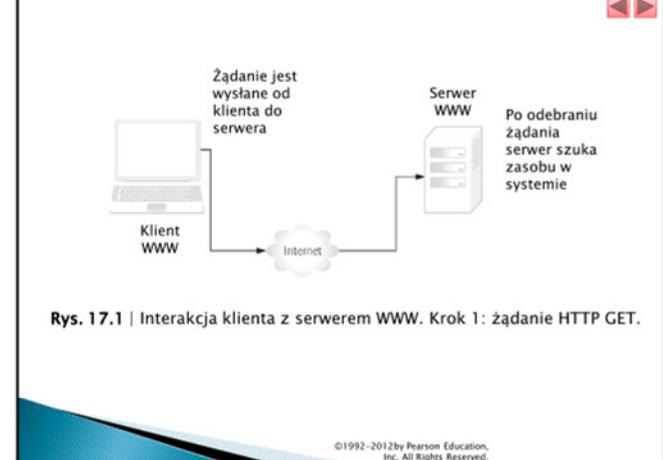
- ▶ Gdy przeglądarka internetowa otrzyma URL strony internetowej, do wysłania żądania używa protokołu HTTP.
- ▶ Metoda HTTP GET wskazuje, że klient chce uzyskać dostęp do zasobu na serwerze.
- ▶ Pozostała część żądania przekazuje ścieżkę dostępu do zasobu (np. dokumentu HTML5), nazwę protokołu oraz jego wersję (HTTP/1.1).
- ▶ Każdemu serwerowi, który rozumie HTTP, można przesłać żądanie GET, a on prawidłowo na nie odpowie.
- ▶ Kod stanu HTTP 200 wskazuje na sukces.
- ▶ Kod stanu 404 informuje klienta, że serwer WWW nie mógł zlokalizować żądanego zasobu.
- ▶ Kompletna lista kodów numerycznych wskazujących na stan transakcji HTTP znajduje się pod adresem: [www.w3.org/Protocols/rfc2616/rfc2616-sec10.html](http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html)

©1992-2012 by Pearson Education,  
Inc. All Rights Reserved.

## 17.2 Transakcje HTTP (cd.)

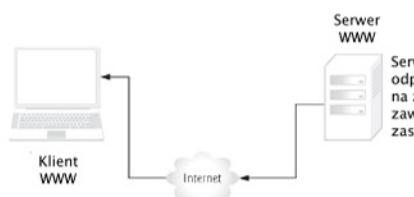
- ▶ Następnie serwer wysyła jeden lub więcej nagłówków HTTP, które dostarczają dodatkowych informacji na temat wysyłanych danych.
- ▶ Standard MIME (Multipurpose Internet Mail Extensions) określa formaty danych, które programy mogą użyć do poprawnej interpretacji danych.
- ▶ Typ MIME `text/plain` wskazuje, że wysyłane dane są tekstem i mogą być bezpośrednio wyświetcone.
- ▶ Typ MIME `image/jpeg` wskazuje, że zawartość jest obrazem JPEG.
- ▶ Gdy przeglądarka otrzymuje informację o takim typie MIME, próbuje wyświetlić obraz.
- ▶ Po nagłówku lub kilku nagłówkach następuje pusta linia, która wskazuje przeglądarce, że serwer skończył wysyłanie nagłówków HTTP.

©1992-2012 by Pearson Education, Inc. All Rights Reserved.



Rys. 17.1 | Interakcja klienta z serwerem WWW. Krok 1: żądanie HTTP GET.

©1992-2012 by Pearson Education, Inc. All Rights Reserved.



Rys. 17.2 | Interakcja klienta z serwerem WWW. Krok 2: odpowiedź HTTP 200 OK.

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

## 17.2 Transakcje HTTP (cd.)

- ▶ Dwa najbardziej popularne typy żądań HTTP
  - Żądanie **GET** zazwyczaj pobiera (pozyskuje) informacje z serwera, takie jak dokument HTML, obraz lub wyniki wyszukiwania na podstawie przedłożonej przez użytkownika frazy wyszukiwania.
  - Żądanie **POST** zazwyczaj zamieszcza (wysyła) dane na serwerze.
  - Popularne zastosowanie żądań **POST** to wysyłanie informacji na serwer, takich jak informacje uwierzytelniania lub dane wprowadzone przez użytkownika do formularza.

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

## 17.2 Transakcje HTTP (cd.)

- ▶ Żądanie HTTP często wysyła dane do programu po stronie serwera, który przetwarza te dane.
- ▶ Żądanie **GET** dołącza te dane do URL w postaci łańcucha zapytania.
- ▶ Do oddzielenia łańcucha zapytania od reszty adresu URL stosuje się znak zapytania (?).
- ▶ Łańcuch zapytania składa się z jednej lub wielu par *nazwa/wartość*. Między nazwą a wartością stosowany jest znak równości (=).
- ▶ Jeśli występuje więcej niż jedna para *nazwa/wartość*, separatorem jest znak **ampersand** (&).

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

## 17.2 Transakcje HTTP (cd.)

- ▶ Żądanie **GET** może być zainicjowane przez kliknięcie kontrolki *submit* formularza HTML, którego atrybut **method** został ustawiony na **get** lub poprzez wpisanie adresu URL (zawierającego łańcuch zapytania) bezpośrednio do paska adresu w przeglądarce.
- ▶ Żądanie **POST** wysyła dane formularza jako wiadomość HTTP, a nie jako część URL.
- ▶ Żądanie **GET** ogranicza długość łańcucha zapytań do określonej liczby znaków.
- ▶ Duże ilości informacji muszą być wysyłane poprzez metodę **POST**.

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

 **Obserwacja inżynierii oprogramowania 17.1**

Dane wysyłane w żądaniu **POST** nie są częścią adresu URL i użytkownik domyślnie ich nie widzi. Jednakże istnieją narzędzia, które mogą ujawnić te dane, więc nie powinno się zakładać, że dane są bezpieczne tylko dlatego, że zostało użyte żądanie **POST**.

©1992-2012 by Pearson Education, Inc.  
All Rights Reserved.

## 17.2 Transakcje HTTP (cd.)

- ▶ Przeglądarki często buforują niedawno wyświetlane strony internetowe, dzięki czemu mogą szybko załadować te strony.
- ▶ Jeśli nie ma żadnych zmian pomiędzy wersją przechowywaną w pamięci a aktualną wersją w sieci WWW, uzyskuje się przyspieszenie przeglądania Internetu.

©1992-2012 by Pearson Education,  
Inc. All Rights Reserved.

## 17.3 Architektura aplikacji wielowarstwowych

- ▶ Aplikacje webowe są często aplikacjami wielowarstwowymi, które dzielą funkcjonalność na oddzielne warstwy.
- ▶ Mimo iż warstwy mogą być zlokalizowane na tym samym komputerze, zazwyczaj są one rozmieszczone na oddzielnych komputerach.
- ▶ **Dolina warstwa** (zwana także warstwą danych lub warstwą informacji) zarządza danymi aplikacji.
- ▶ **Środkowa warstwa** implementuje logikę biznesową, sterowania i prezentacji w celu kontrolowania interakcji pomiędzy klientami aplikacji i jej danymi.

©1992-2012 by Pearson Education,  
Inc. All Rights Reserved.

## 17.3 Architektura aplikacji wielowarstwowych (cd.)

- ▶ **Logika biznesowa** w środkowej warstwie wymusza zasady biznesowe i zapewnia, że dane są wiarygodne, zanim aplikacja serwera zaktualizuje bazę danych lub zaprezentuje dane użytkownikom.
- ▶ Zasady biznesowe dyktują, w jaki sposób klient może dostać się do danych i jak aplikacje przetwarzają dane.
- ▶ **Górna warstwa** (warstwa klienta) jest interfejsem użytkownika aplikacji.
- ▶ W odpowiedzi na akcje użytkownika, warstwa klienta wchodzi w interakcję ze środkową warstwą, aby wysłać żądania i aby pozyskać dane z warstwy informacyjnej.
- ▶ Następnie warstwa klienta wyświetla pozyskane dane użytkownikowi.
- ▶ Warstwa klienta nigdy nie komunikuje się bezpośrednio z warstwą informacji.

©1992-2012 by Pearson Education,  
Inc. All Rights Reserved.

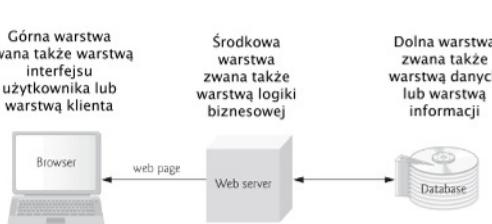


Diagram illustrating the three-layer architecture:

- Górna warstwa** (zwana także warstwą interfejsu użytkownika lub warstwą klienta): Browser
- Środkowa warstwa** (zwana także warstwą logiki biznesowej): Web server
- Dolina warstwa** (zwana także warstwą danych lub warstwą informacji): Database

The diagram shows the flow of data from the Browser through the Web server to the Database.

©1992-2012 by Pearson Education,  
Inc. All Rights Reserved.

Rys. 17.3 | Trójwarstwowa architektura.

## 17.4 Programowanie po stronie klienta a programowanie po stronie serwera

- ▶ Programowanie po stronie klienta może być użyte do
  - validacji danych wprowadzonych przez użytkownika,
  - usprawnienia interakcji z przeglądarką,
  - poprawienia wyglądu stron internetowych,
  - zainicjowania komunikacji pomiędzy przeglądarką a serwerem WWW.
- ▶ Programowanie po stronie klienta ma ograniczenia, takie jak zależność od przeglądarki.
- ▶ Przeglądarka lub tzw. *scripting host* musi obsługiwać język skryptowy i jego możliwości.
- ▶ Kod programów po stronie klienta może zostać wyświetlony przez klienta za pomocą odpowiedniej opcji w przeglądarce (wyświetlanie źródła).

©1992-2012 by Pearson Education,  
Inc. All Rights Reserved.

## 17.4 Programowanie po stronie klienta a programowanie po stronie serwera (cd.)

- ▶ Poufne informacje, takie jak hasła lub inne dane identyfikujące użytkownika, nie powinny być przechowywane lub walidowane po stronie klienta.
- ▶ Umieszczanie dużej ilości kodu JavaScript po stronie klienta może doprowadzić do problemów z bezpieczeństwem aplikacji webowych.
- ▶ Kod wykonywany na serwerze często generuje różne odpowiedzi dla klientów.
- ▶ Języki programowania po stronie serwera mają szerszy zakres możliwości programistycznych niż ich odpowiedniki po stronie klienta.

©1992-2012 by Pearson Education,  
Inc. All Rights Reserved.

## 17.5 Uzyskiwanie dostępu do serwerów WWW

- ▶ Aby żądać dokumenty z serwerów WWW, użytkownicy muszą znać nazwy hostów, na których znajduje się oprogramowanie serwera WWW.
- ▶ Użytkownicy mogą żądać dokumentów z lokalnych lub z odległych serwerów WWW.
- ▶ Dostęp do lokalnych serwerów WWW można uzyskać poprzez nazwę komputera lub nazwę **localhost**, która odnosi się do lokalnej maszyny i zazwyczaj tłumaczona jest na adres IP **127.0.0.1** (znany także jako pętla zwrotna).

©1992-2012 by Pearson Education,  
Inc. All Rights Reserved.

## 17.6 Instalacja Apache, MySQL i PHP

- ▶ Serwer HTTP Apache, utrzymywany przez Apache Software Foundation, jest aktualnie najbardziej popularnym serwerem WWW. Jest to oprogramowanie open source, na platformy UNIX, Linux, Mac OS X, Windows i wiele innych.
- ▶ MySQL jest najbardziej popularnym systemem zarządzania bazą danych (open source). Przeznaczony jest również na platformy: Linux, Mac OS X i Windows i inne.
- ▶ Serwer HTTP Apache, MySQL i PHP mogą być osobno pobrane i zainstalowane, ale to wymaga także dodatkowej konfiguracji.

©1992-2012 by Pearson Education,  
Inc. All Rights Reserved.

## 17.6 Instalacja Apache, MySQL i PHP

- ▶ Dla uproszczenia używany będzie zintegrowany instalator XAMPP dostarczony przez stronę Apache Friends ([www.apachefriends.org](http://www.apachefriends.org)).
- ▶ Na stronie <https://www.apachefriends.org/pl/index.html> można wybrać instalator do określonej platformy. Obecnie w skład XAMPP zamiast MySQL wchodzi Maria DB – fork (odgałęzienie) MySQL w założeniu zgodny z tą bazą.
- ▶ Można także używać PHP z Microsoft IIS Express i SQL Server Express: <https://docs.microsoft.com/en-us/iis/application-frameworks/scenario-build-a-php-website-on-iis/configure-a-php-website-on-iis>

©1992-2012 by Pearson Education,  
Inc. All Rights Reserved.

### 17.6.2 Uruchamianie XAMPP

#### Windows

- ▶ Należy uruchomić plik `xampp_start.exe` w folderze, w którym zainstalowano XAMPP (domyślnie `c:\xampp`). W przypadku potrzeby zatrzymania serwerów (np. aby wyłączyć komputer), należy uruchomić plik `xampp_stop.exe` w tym samym folderze.

#### Mac OS X

- ▶ Należy udać się do folderu Applications (lub folderu, w którym zainstalowano XAMPP), następnie otworzyć folder XAMPP i uruchomić XAMP Control.app. Należy kliknąć przycisk Start w panelu sterowania, aby uruchomić serwery. Aby zatrzymać serwery, należy kliknąć przycisk Stop.

#### Linux

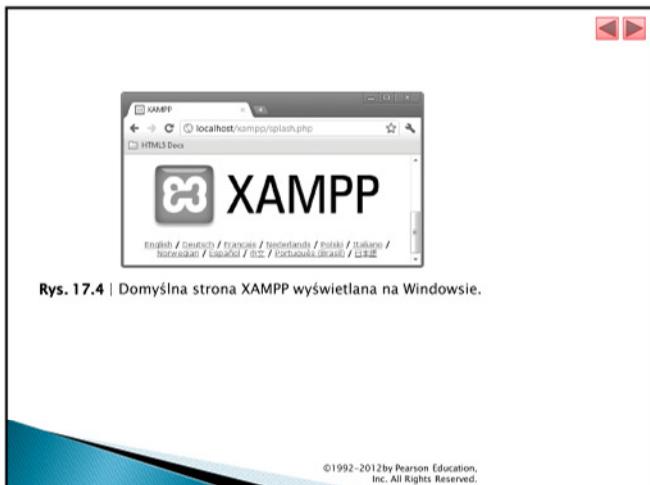
- ▶ Należy w trybie terminalowym (shell) wprowadzić komendę `/opt/lampp/lampp start`
- ▶ Aby zatrzymać serwery, należy podać komendę `/opt/lampp/lampp stop`

©1992-2012 by Pearson Education,  
Inc. All Rights Reserved.

### 17.6.3 Testowanie konfiguracji

- ▶ Po uruchomieniu serwerów, można otworzyć przeglądarkę internetową na tym samym komputerze i wprowadzić adres `http://localhost/`, aby upewnić się, że serwer WWW działa prawidłowo.
- ▶ Jeśli ukaże się strona podobna do pokazanej na rysunku 17.4, wszystko jest gotowe do pracy.

©1992-2012 by Pearson Education,  
Inc. All Rights Reserved.



Rys. 17.4 | Domyślna strona XAMPP wyświetlaną na Windowsie.

## 19.1 Wstęp

- ▶ PHP lub PHP: Hypertext Preprocessor, stał się jednym z najbardziej popularnych języków skryptowych działających po stronie serwera i jest używany do tworzenia dynamicznych stron internetowych.
- ▶ PHP jest niezależny od platformy i dostępny jako *open source* — istnieją implementacje dla wszystkich najważniejszych systemów: UNIX, Linux, Mac i Windows. PHP współpracuje także z wieloma bazami danych.

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

## 19.2 Prosty program PHP

- ▶ Siłą WWW jest nie tylko udostępnianie zawartości użytkownikom, ale też odpowiadanie na żądania użytkowników i generowanie stron internetowych z dynamiczną zawartością.
- ▶ Kod PHP jest osadzony bezpośrednio w dokumentach HTML5, ale segmenty tych skryptów są interpretowane przez serwer przed dostarczeniem dokumentu do klienta.
- ▶ Nazwy plików skryptów PHP mają rozszerzenie .php.
- ▶ Wprawdzie PHP może być używane z linii poleceń, jednak do skorzystania ze wszystkich zalet języka skryptowego konieczny jest serwer internetowy.
- ▶ W PHP, kod umieszczony jest pomiędzy skryptowymi ogranicznikami <?php i ?>. Kod PHP może być umieszczony w każdym miejscu w znaczniku HTML5, o ile znajduje się w tych ogranicznikach.

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

## 19.2 Prosty program PHP (cd.)

- ▶ Zmienne poprzedzone są znakiem \$ i są inicjowane przy pierwszym napotkaniu podczas interpretacji kodu.
- ▶ Polecenia PHP zakończone są średnikiem (;).
- ▶ Pojedyncza linia komentarza zaczyna się dwoma ukośnikami (//) lub znakiem hash (#). Tekst po prawej stronie od tych znaków jest ignorowany przez interpreter. Komentarze składające się z wielu linii zaczynają się od ogranicznika /\* i kończą ogranicznikiem \*/.
- ▶ Kiedy zmienne umieszczone są w cudzysłowie (""), PHP interpoluje zmienną. Innymi słowy, PHP zamienia nazwę zmiennej na jej wartość.
- ▶ Wszystkie operacje wymagające interpolacji PHP wykonywane są na serwerze przed wysłaniem dokumentu HTML5 do klienta.
- ▶ Zmienne PHP są typowane dynamicznie – mogą zawierać wartości różnych typów w różnych momentach.

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

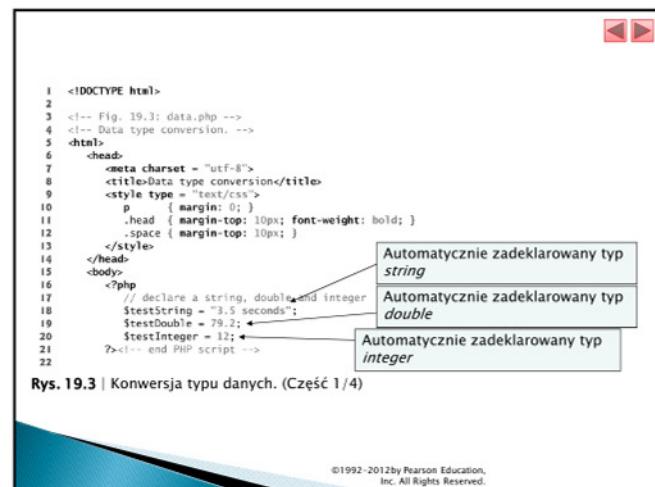
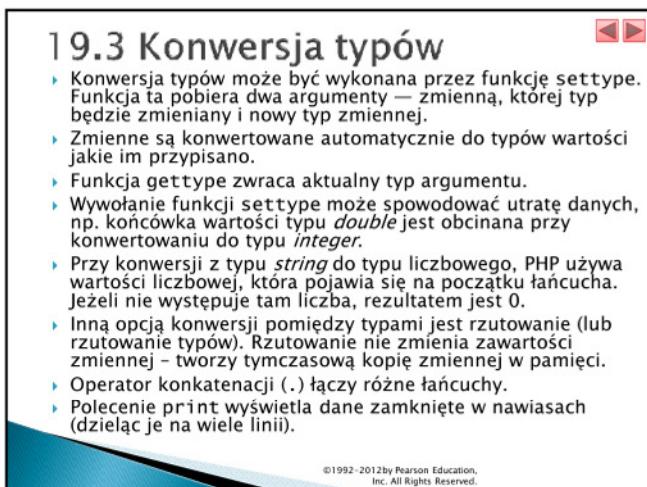
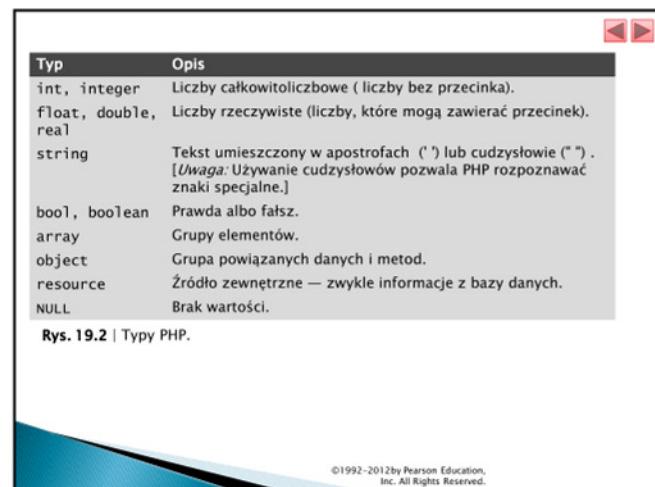
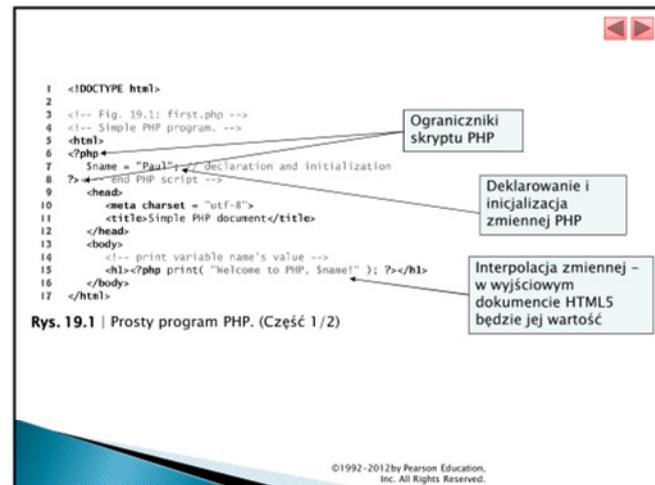
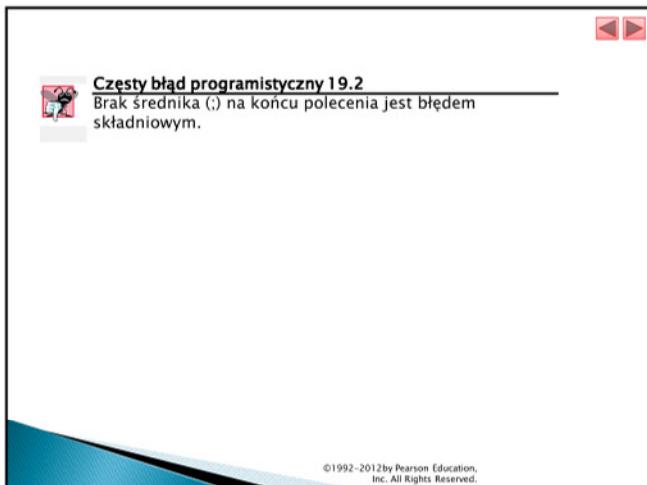
**Częsty błąd programistyczny 19.0**  
W przypadku braku poprzedzenia nazwy zmiennej znakiem \$ występuje błąd składni.

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

### Częsty błąd programistyczny 19.1

PHP jest wrażliwe na wielkość liter w nazwach zmiennych. Zamiana wielkości liter w odniesieniach do zmiennych spowoduje błąd logiczny, ponieważ skrypt utworzy nową zmienną dla każdej nazwy, która różni się od nazwy już zadeklarowanej zmiennej.

©1992-2012 by Pearson Education, Inc. All Rights Reserved.



```

23 <!-- print each variable's value and type -->
24 <p class = "head">Original values:</p>
25 <p>
26     print( "<p>$testString is a(n) " . gettype( $testString ) );
27     . "</p>" );
28     print( "<p>$testDouble is a(n) " . gettype( $testDouble ) );
29     . "</p>" );
30     print( "<p>$testInteger is a(n) " . gettype( $testInteger ) );
31     . "</p>" );
32 ?<!-- end PHP script -->
33 <p class = "head">Converting to other data types:</p>
34 <p>
35 // call function settype to convert variable
36 // testString to different data types
37 print( "<p>$testString " );
38 settype( $testString, "double" );
39 print( "<p>$testString is $testString</p>" );
40 print( "<p>$testString " );
41 settype( $testString, "integer" );
42 print( " as an integer is $testString</p>" );
43 settype( $testString, "string" );
44 print( "<p class = 'space'>Converting back to a string results in
45 $testString</p>" );
46

```

Rys. 19.3 | Konwersja typu danych. (Część 2/4)

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

```

47 // use type casting to cast variables to a different type
48 $data = "98.6 degrees";
49 print( "<p class = 'space'>Before casting: $data is a " .
50     gettype( $data ) . "</p>" );
51 print( "<p class = 'space'>Using type casting instead:</p>" );
52 "as a double: " . (double) $data . "</p>" ,
53 "as an integer: " . (integer) $data . "</p>" ;
54 print( "<p class = 'space'>After casting: $data is a " .
55     gettype( $data ) . "</p>" );
56 ?><!-- end PHP script -->
57 </body>
58 </html>

```

Rys. 19.3 | Konwersja typu danych. (Część 3/4)

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

Original values:  
3.5 seconds is a(n) string  
79.2 is a(n) double  
12 is a(n) integer

Converting to other data types:  
3.5 seconds as a double is 3.5  
3.5 as an integer is 3

Converting back to a string results in 3

Before casting: 98.6 degrees is a string

Using type casting instead:  
as a double: 98.6  
as an integer: 98

After casting: 98.6 degrees is a string

Rys. 19.3 | Konwersja typu danych. (Część 4/4)

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

**Wskazówka dotycząca zapobiegania błędom 19.1**

Funkcja print może być używana do wyświetlania wartości zmiennych w różnych miejscach podczas wykonywania programu. Często pomaga to w debugowaniu skryptu.

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

## 19.4 Operacje arytmetyczne

- Funkcja define tworzy stałe. Pobiera dwa argumenty — nazwę i wartość stałej. Opcjonalny, trzeci argument jest typu boolean i określa czy stała będzie wrażliwa na wielkość liter. Domyślnie, stałe są wrażliwe na wielkość liter.
- Niezainicjowana stała ma wartość undef, która może mieć różne znaczenia w zależności od kontekstu. W kontekście numerycznym wartość ta wynosi 0. W przypadku string oznacza łańcuch pusty ("").
- Słowa kluczowe nie mogą być używane jako nazwy.

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

**Czasty błąd programistyczny 19.3**

Przypisanie wartości do stałej po jej zadeklarowaniu powoduje błąd składni.

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

```

1 <!DOCTYPE html>
2
3 <!-- Fig. 19.4: operators.php -->
4 <!-- Using arithmetic operators. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <style type = "text/css">
9       p { margin: 0; }
10    </style>
11    <title>Using arithmetic operators</title>
12  </head>
13  <body>
14    <?php
15      $a = 5;
16      print( "<p>The value of variable a is $a</p>" );
17
18      // define constant VALUE
19      define( "VALUE", 5 ); ← Tworzenie stałej o
20                                nazwie VALUE o
21                                wartości 5
22
23      // add constant VALUE to variable $a
24      $a += VALUE;
25      print( "<p>Variable a after adding constant VALUE is $a</p>" );
26

```

**Rys. 19.4 | Użycie operatorów arytmetycznych. (Część 1/4)**

©1992-2012 by Pearson Education,  
Inc. All Rights Reserved.

```

26      // multiply variable $a by 2
27      $a *= 2; ← Równoznaczne wyrażeniu $a = $a * 2
28      print( "<p>Multiplying variable a by 2 yields $a</p>" );
29
30      // test if variable $a is less than 50
31      if ( $a < 50 )
32        print( "<p>Variable a is less than 50</p>" );
33
34      // add 40 to variable $a
35      $a += 40;
36      print( "<p>Variable a after adding 40 is $a</p>" );
37
38      // test if variable $a is 50 or less
39      if ( $a < 51 ) ← Użycie operatora
40        print( "<p>Variable a is still 50 or less</p>" );
41      elseif ( $a < 101 ) // $a >= 51 and <= 100
42        print( "<p>Variable a is now between 50 and 100,
43           inclusive</p>" );
44      else // $a > 100
45        print( "<p>Variable a is now greater than 100</p>" );
46
47      // print an uninitialized variable
48      print( "<p>Using a variable before initializing
49           $nothing</p>" ); // nothing evaluates to ""
50

```

**Rys. 19.4 | Użycie operatorów arytmetycznych. (Część 2/4)**

©1992-2012 by Pearson Education,  
Inc. All Rights Reserved.

```

50
51      // add constant VALUE to an uninitialized variable
52      $test = $num + VALUE; // num evaluates to 0 ← Niezainicjowana
53      print( "<p>An uninitialized variable plus constant
54                                VALUE yields $test</p>" );
55
56      // add a string to an integer
57      $str = "3 dollars"; ← Na potrzeby tej
58      $a += $str;          operacji zmieniona
59      print( "<p>Adding a string to variable a yields $a</p>" );
60
61  </body>
62  </html>

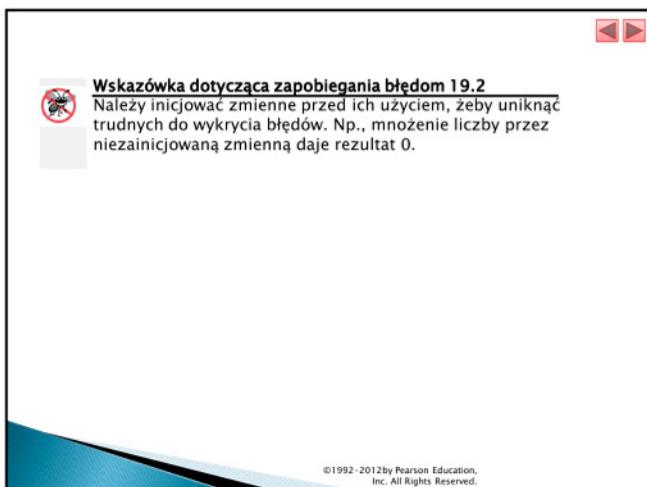
```

**Rys. 19.4 | Użycie operatorów arytmetycznych. (Część 3/4)**

©1992-2012 by Pearson Education,  
Inc. All Rights Reserved.

**Rys. 19.4 | Użycie operatorów arytmetycznych. (Część 4/4)**

©1992-2012 by Pearson Education,  
Inc. All Rights Reserved.



PHP keywords				
abstract	and	array	as	break
case	catch	class	clone	const
continue	declare	default	do	else
elseif	enddeclare	endfor	endforeach	endif
endswitch	endwhile	extends	final	for
foreach	function	global	goto	if
implements	interface	instanceof	namespace	new
or	private	protected	public	static
switch	throw	try	use	var
while	xor			

**Rys. 19.5 | Słowa kluczowe PHP.**

©1992-2012 by Pearson Education,  
Inc. All Rights Reserved.

## Mapa pierwszeństwa operatorów

▶ Tabele na kolejnych slajdach prezentują listę operatorów PHP uporządkowanych wg pierwszeństwa wykonania w kolejności malejącej.

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

Operator	Typ	Powiązanie
New	konstruktor	brak
[]	Indeks	Od prawej do lewej
~	bitowe not	Od prawej do lewej
!	not	Od prawej do lewej
++	zwiększenie	
--	zwiększenie	
-	jednoargumentowa zmiana znaku	
@	kontrola błędów	
*	mnożenie	Z lewej do prawej
/	dzielenie	Z lewej do prawej
%	reszta z dzielenia	Z lewej do prawej
+	dodawanie	Z lewej do prawej
-	odejmowanie	Z lewej do prawej
.	konkatencja	

Rys. 19.6 | Operatory PHP: pierwszeństwa i powiązania.  
(Część 1/3)

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

Operator	Typ	Powiązanie
<<	Bitowe przesunięcie w lewo	Z lewej do prawej
>>	Bitowe przesunięcie w prawo	
<	Mniejszy niż	brak
>	Większy niż	
<=	Mniejszy równy	
>=	Większy równy	
==	równy	brak
!=	nierówny	
==	identyczny	
!=	nieidentyczny	
&	Bitowe AND	Z lewej do prawej
^	Bitowe XOR	Z lewej do prawej
	Bitowe OR	Z lewej do prawej
&&	Logiczne AND	Z lewej do prawej
	Logiczne OR	Z lewej do prawej

Rys. 19.6 | Operatory PHP: pierwszeństwa i powiązania.  
(Część 2/3)

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

Operator	Typ	Powiązanie
=	przypisanie	Z lewej do prawej
+=	przypisanie dodawania	Z lewej do prawej
-=	przypisanie odejmowania	Z lewej do prawej
*=	przypisanie mnożenia	Z lewej do prawej
/=	przypisanie dzielenia	Z lewej do prawej
&=	przypisanie bitowego AND	Z lewej do prawej
=	przypisanie bitowego OR	Z lewej do prawej
^=	przypisanie bitowej różnicy symetrycznej	Z lewej do prawej
.=	przypisanie konkatencji t	Z lewej do prawej
<<=	przyp. bitowego przesunięcia w lewo	Z lewej do prawej
>>=	przyp. bitowego przesunięcia w prawo	Z lewej do prawej
and	logiczne AND	Z lewej do prawej
xor	bitowa różnica symetryczna	Z lewej do prawej
or	logiczne OR	Z lewej do prawej
,	lista	Z lewej do prawej

Rys. 19.6 | Operatory PHP: pierwszeństwa i powiązania.  
(Część 3/3)

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

## 19.5 Tablice w PHP

- ▶ PHP umożliwia przechowywanie danych w tablicach. Tablice są podzielone na elementy, które przechowują pojedyncze zmienne. Nazwy tablic, tak jak innych zmiennych, zaczynają się od symbolu \$.
- ▶ Odwołanie do elementu tablicy wykonywane jest przez podanie nazwy tablicy oraz jego indeksu w nawiasach kwadratowych ([]).
- ▶ Jeżeli wartość jest przypisywana do tablicy, która nie istnieje, skrypt utworzy nową tablicę. Przypisywanie wartości do elementu, który nie istnieje w tablicy powoduje dodanie nowego elementu na końcu tablicy.
- ▶ Funkcja **count** zwraca liczbę elementów w tablicy.
- ▶ Funkcja **array** tworzy tablicę zawierającą elementy podane jako argumenty funkcji. Pierwszy argument z listy jest przechowywany jako pierwszy element w tablicy (o indeksie 0), drugi argument jest drugim elementem tablicy.

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

19.5 Tablice w PHP (cd.)
▶ Tablice z indeksowaniem nienumerycznym nazywane są tablicami asocjacyjnymi. Takie tablice tworzy się wykorzystując operator =>, gdzie wartość po lewej stronie operatora jest indeksem, a wartość po prawej stronie jest wartością elementu tablicy.
▶ PHP udostępnia funkcje umożliwiające iteracje po elementach tablic. Każda tablica ma wbudowany wewnętrzny wskaźnik, który wskazuje na wybrany element.
▪ Funkcja <b>reset</b> ustawia ten wskaźnik na pierwszy element.
▪ Funkcja <b>key</b> zwraca indeks elementu obecnie wskazywanego.
▪ Funkcja <b>next</b> przemieszcza wskaźnik na następny element.
▶ Polecenie <b>foreach</b> służy do iteracji po tablicach. Po nim umieszczone jest słowo kluczowe <b>as</b> , a następnie dwie zmienne – pierwsza jest indeksem, druga jest wartością elementu. Jeżeli po as, umieszczone jest tylko jedna zmienna, przypisywana do niej jest wartość elementu

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

```

1 <!DOCTYPE html>
2
3 <!-- Fig. 19.7: arrays.php -->
4 <!-- Array manipulation. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Array manipulation</title>
9     <style type = "text/css">
10       p { margin: 0; }
11       .head { margin-top: 10px; font-weight: bold; }
12   </head>
13   <body>
14     <?php
15
16       // create array $first
17       print( "<p class = 'head'>Creating the first array</p>" );
18       $first[ 0 ] = "zero";
19       $first[ 1 ] = "one";
20       $first[ 2 ] = "two";
21       $first[] = "three";
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47

```

**Automatyczne tworzenie tablicy \$first**

**Ustawienie wartości pierwszego elementu \$first na "zero"**

**"three" jest dodane na koniec tablicy \$first**

Rys. 19.7 | Manipulowanie tablicami. (Część 1/4)

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

```

23
24   // print each element's index and value
25   for ( $i = 0; $i < count( $first ); ++$i )
26     print( "Element $i is $first[$i]</p>" );
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47

```

**Zwrota liczbę elementów w tablicy**

**Funkcja array tworzy tablice \$second z elementami podanymi jako argumenty**

**Tworzenie tablicy asocjacyjnej \$third**

**Ustawia wskaźnik na pierwszy element \$third**

**Zwrota indeks aktualnie wskazywanego elementu**

**Przemieszcza wskaźnik na następny element i zwraca jego wartość**

Rys. 19.7 | Manipulowanie tablicami. (Część 2/4)

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

```

48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64

```

**Użycie operatora => do inicjalizacji elementu o indeksie "January" i wartości "first"**

**Iteruje po każdym elemencie w tablicy \$fourth**

**Przechowuje aktualną wartość indeksu**

**Przechowuje wartość elementu**

Rys. 19.7 | Manipulowanie tablicami. (Część 3/4)

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

Rys. 19.7 | Manipulowanie tablicami. (Część 4/4)

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

## 19.6 Porównywanie łańcuchów

- Wyrażenia regularne, to seria znaków użytych do przypasowania do szablonu łańcucha, plików tekstowych i baz danych.
- Wiele zadań przetwarzania łańcuchów może być wykonanych poprzez użycie operatorów równości i relacji (==, !=, <, <=, >, >=).
- Funkcja **strcmp** porównuje dwa łańcuchy. Funkcja zwraca -1, jeżeli pierwszy argument jest alfabetycznie wcześniejszy, 0 jeżeli są takie same, 1 jeżeli pierwszy element jest alfabetycznie drugi.

Rys. 19.8 | Operatory porównywania łańcuchów. (Część 1/3)

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

```

1 <!DOCTYPE html>
2
3 <!-- Fig. 19.8: compare.php -->
4 <!-- Using the string-comparison operators. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>String Comparison</title>
9     <style type = "text/css">
10       p { margin: 0; }
11     </style>
12   </head>
13   <body>
14     <?php
15
16       // create array fruits
17       $fruits = array("apple", "orange", "banana" );
18
19
20
21
22
23
24

```

**Sprawdza czy i-ty element tablicy fruits poprzedza łańcuch banana**

Rys. 19.8 | Operatory porównywania łańcuchów. (Część 1/3)

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

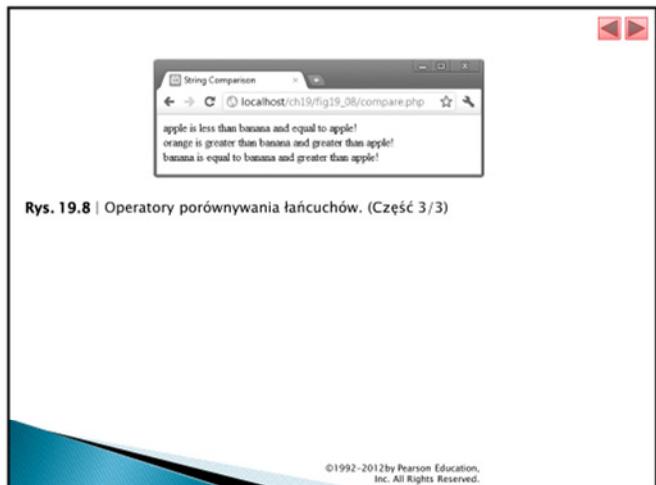
```

25      elseif (strcmp( $fruits[ $i ], "banana" ) > 0 )
26          print("<p>" . $fruits[ $i ] . " is greater than banana ");
27      else
28          print "<p>" . $fruits[ $i ] . " is equal to banana " ;
29
30      // use relational operators to compare each element
31      // to string "apple"
32      if ( $fruits[ $i ] < "apple" )
33          print("and less than apple!</p>");
34      elseif ( $fruits[ $i ] > "apple" )
35          print("and greater than apple!</p>");
36      elseif ( $fruits[ $i ] == "apple" )
37          print("and equal to apple!</p>");
38    } // end for
39  ?<!-- end PHP script -->
40 </body>
41 </html>

```

Użycie operatorów porównania elementu tablicy fruits z łańcuchem apple

Rys. 19.8 | Operatory porównywania łańcuchów. (Część 2/3)



Rys. 19.8 | Operatory porównywania łańcuchów. (Część 3/3)

### 19.7 Przetwarzanie łańcuchów i wyrażenia regularne

- Funkcja **preg\_match** używa wyrażeń regularnych (**Perl-Compatible Regular Expressions – PCRE**) do wyszukiwania łańcucha według wyspecyfikowanego schematu (wzorca).
- Jeżeli schemat zostanie znaleziony przy użyciu **preg\_match**, funkcja zwraca **długość dopasowanego łańcucha** (liczbę dodatnią) — może być to interpretowane jako logiczne **true**.
- Funkcja **preg\_match** ma dwa argumenty: schemat wyrażenia regularnego oraz łańcuch, w którym będzie wyszukane to wyrażenie.

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

### 19.7 Przetwarzanie łańcuchów i wyrażenia regularne

- Wyrażenia regularne muszą być ujęte w ograniczniki, np. (/).
  - Wyrażenia regularne mogą zawierać **metaznaki**, które precyzują schemat. Przykładowo, metaznak karetki (^) określa początek łańcucha, dolara (\$) – koniec łańcucha, kropki (.) – pojedynczy znak.
  - Wyrażenie klamrowe**, to lista znaków umieszczonych w nawiasach kwadratowych ([]). Znak będzie musiał pasować do jednego z listy. Zakres może być wyrażony też za pomocą myślnika (-) przez podanie początku i końca zakresu.
- ©1992-2012 by Pearson Education, Inc. All Rights Reserved.

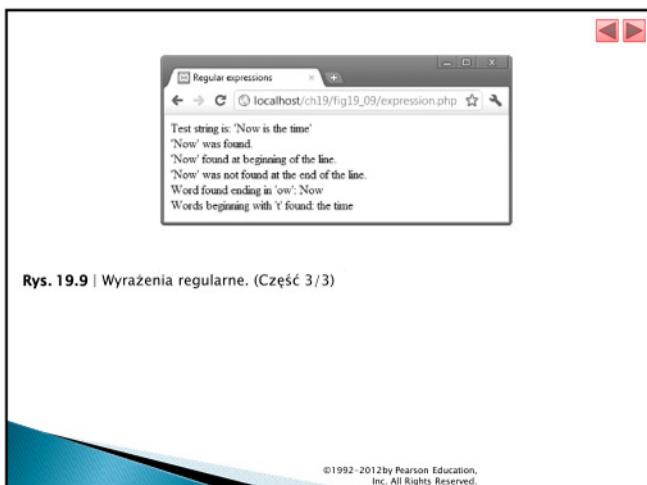
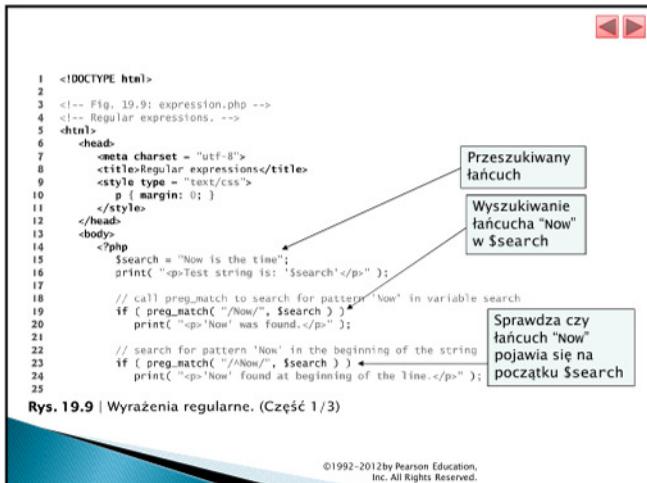
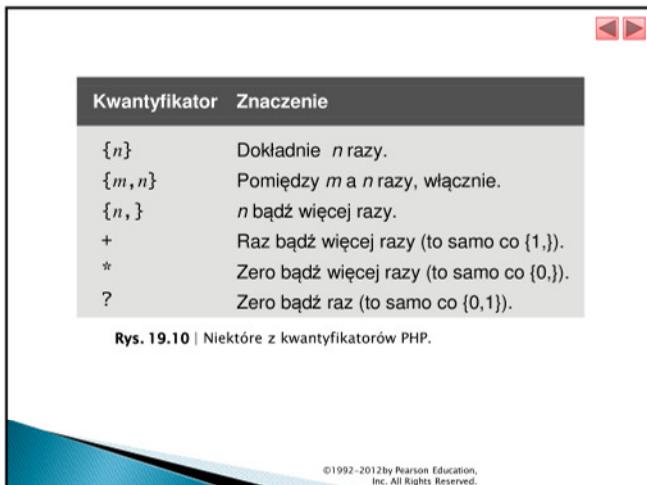
Grupy znaków	Opis
alnum	Znaki alfa numeryczne (tj., litery [a-zA-Z] lub cyfry [0-9]).
alpha	Znaki występujące w słowach (tj., litery [a-zA-Z]).
digit	Cyfry.
space	Znaki białe.
lower	Male litery.
upper	Wielkie litery.

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

Rys. 19.11 | Niektóre z grup znaczników PHP.

### 19.7 Przetwarzanie łańcuchów i wyrażenia regularne (cd.)

- Metaznak \b przed i po nawiasie wskazuje początek i koniec słowa – jest to sposób na dopasowanie tylko całych słów, np. /\b([alpha]\*)\b/
  - Nieczułość na wielkość znaków osiąga się przez: /.../i
  - Kwantyfikatory** są używane w wyrażeniach regularnych do określenia, jak często dany znak lub zestaw znaków może pojawić się we wzorcu.
  - Opcjonalny, trzeci argument funkcji **preg\_match** jest tablicą, w której przechowywane są łańcuchy pasujące do umieszczonej w nawiasach podrzędnych wyrażeń regularnych. Pierwszy element tablicy przechowuje łańcuch pasujący do całego wzorca; pozostałe elementy przechowują kolejne części indeksowane od lewej do prawej.
- ©1992-2012 by Pearson Education, Inc. All Rights Reserved.



Rys. 19.9 | Wyrażenia regularne. (Część 3/3)

## 19.7 Przetwarzanie łańcuchów i wyrażenia regularne (cd.)

- Aby znaleźć wiele wystąpień danego wzorca, należy wiele razy wywołać funkcję **preg\_match**. Przed każdym kolejnym wywołaniem należy usunąć wynik. Wykonuje się to wykorzystując funkcję **preg\_replace**.
- Funkcja **preg\_replace** ma trzy argumenty: wzorzec, nowy łańcuch, łańcuch, w którym ma być odnaleziony wzorzec i zastąpiony nowym łańcuchem. Wynikiem jest zmodyfikowany łańcuch.
- Klasy znaków umieszcza się pomiędzy ogranicznikami **[ : i : ]**. Zastosowanie podwójnych ograniczników **[[:i:]]** powoduje, że wyrażenie regularne dopasowuje wszystkie pojedyncze znaki należące do klasy.

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

Sprawdza czy łańcuch "Now" pojawia się na końcu \$search

```

26 // search for pattern 'Now' at the end of the string
27 if ( !preg_match( "/Now$/i", $search ) )
28   print( "<p>'Now' was not found at the end of the line.</p>" );
29
30 // search for any word ending in 'ow'
31 if ( preg_match( "/\b([a-zA-Z]ow)\b/i", $search, $match ) )
32   print( "<p>Word found ending in 'ow': " .
33         $match[ 1 ] . "</p>" );
34
35 // search for any words beginning with 't'
36 if ( preg_match( "/(p|Words) beginning with 't' found: " ) );
37
38 while ( preg_match( "/\b(t|[[:alpha:]]+)\b/i", $search, $match ) )
39 {
40   print( $match[ 1 ] . " " );
41
42   // remove the first occurrence of a word beginning
43   // with 't' to find other instances in the string
44   $search = preg_replace( "/^" . $match[ 1 ] . "/", "", $search );
45 } // end while
46
47 print( "</p>" );
48 //-- end PHP script -->
49 </body>
50 </html>

```

Wyszukanie słów kończących się na "ow" (bez rozróżniania wielkości liter) i przechowanie ich w tablicy \$match

Wypisanie wyszukanych słów zaczynających się od "t" i przechowanie ich w tablicy \$match

Zastąpienie słowa wyszukanego przez funkcję **preg\_replace** pustym łańcuchem. W ten sposób kolejne słowa pasujące do wzorca będą zawsze dostępne w \$match[ 1 ].

Rys. 19.9 | Wyrażenia regularne. (Część 2/3)

## 19.8 Przetwarzanie formularzy i logika biznesowa

- Tablice **\$\_GET** i **\$\_POST** otrzymują informacje wysypane do serwera przez żądania HTTP **get** i **post**.
- Użycie **method = "post"** dołącza dane z formularza do ciała żądania HTTP. URL żadanego zasobu znajduje się w linii inicjalnej żądania. Skrypt (będący żdanym zasobem) ulokowany na serwerze webowym zajmuje się zazwyczaj przetwarzaniem danych z formularza.

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

**Nazwa zmiennej Opis**

<b>\$_SERVER</b>	Dane dotyczące obecnie uruchomionego serwera.
<b>\$_ENV</b>	Dane dotyczące środowiska klienta.
<b>\$_GET</b>	Dane wysłane na serwer przez żądanie get.
<b>\$_POST</b>	Dane wysłane na serwer przez żądanie post.
<b>\$_COOKIE</b>	Dane przechowywane jako ciasteczka na komputerze klienta.
<b>\$_GLOBALS</b>	Tablica zawierająca wszystkie zmienne globalne.

Rys. 19.12 | Niektóre z tablic superglobalnych.

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

```

1 <!DOCTYPE html>
2 <!-- Fig. 19.13: form.html -->
3 <!-- HTML form for gathering user input. -->
4 <html>
5   <head>
6     <meta charset = "utf-8">
7     <title>Sample Form</title>
8     <style type = "text/css">
9       label { width: 5em; float: left; }
10      </style>
11    </head>
12    <body>
13      <h2>Registration Form</h2>
14      <p>Please fill in all fields and click Register.</p>
15
16      <!-- post form data to form.php -->
17      <form method = "post" action = "form.php">
18        <h2>User Information</h2>
19
20        <!-- create four text boxes for user input -->
21        <div><label>First name:</label>
22          <input type = "text" name = "fname"></div> ←
23        <div><label>Last name:</label>
24          <input type = "text" name = "lname"></div> ←

```

**Dodanie danych z formularza do żądania przeglądarki. Dane zostaną przesłane w ciele żądania, a nie w URL.**

**Dane formularza zostaną wysłane do form.php, w celu ich przetworzenia.**

**Tworzy pole formularza**

Rys. 19.13 | Formularz HTML5 do zebrania danych wprowadzonych przez użytkownika. (Część 1/4)

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

```

25      <input type = "text" name = "lname"></div>
26      <div><label>Email:</label>
27        <input type = "text" name = "email"></div>
28      <div><label>Phone:</label>
29        <input type = "text" name = "phone"
30          placeholder = "(555) 555-5555"></div>
31
32
33      <h2>Publications</h2>
34      <p>Which book would you like information about?</p>
35
36      <!-- create drop-down list containing book names -->
37      <select name = "book"> ←
38        <option>Internet and WWW How to Program</option>
39        <option>C++ How to Program</option>
40        <option>Java How to Programs</option>
41        <option>Visual Basic How to Program</option>
42      </select>
43
44      <h2>Operating System</h2>
45      <p>Which operating system do you use?</p>
46
47      <!-- create five radio buttons -->
48      <p><input type = "radio" name = "os" value = "Windows" checked>Windows
49
```

**Tworzy rozwijaną listę z nazwami książek**

**Tworzy przyciski opcji z początkowo wybranym „Windows”**

Rys. 19.13 | Formularz HTML5 do zebrania danych wprowadzonych przez użytkownika. (Część 2/4)

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

```

50      <input type = "radio" name = "os" value = "Mac OS X">Mac OS X
51      <input type = "radio" name = "os" value = "Linux">Linux
52      <input type = "radio" name = "os" value = "Other">Other</p>
53
54      <!-- create a submit button -->
55      <p><input type = "submit" name = "submit" value = "Register"></p>
56
57  </body>
58 </html>

```

Rys. 19.13 | Formularz HTML5 do zebrania danych wprowadzonych przez użytkownika. (Część 3/4)

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

Formularz jest wypełniony niepoprawnym numerem telefonu.

Rys. 19.13 | Formularz HTML5 do zebrania danych wprowadzonych przez użytkownika. (Część 4/4)

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

**Dobry zwyczaj programistyczny 19.1**

Należy używać nazw, określających znaczenie obiektów HTML5 dla pól formularza. Sprawi to, że skrypt PHP, który otrzyma te dane stanie się łatwiejszy do zrozumienia.

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

## 19.8 Przetwarzanie formularzy i logika biznesowa (cd.)

- Logika biznesowa, albo reguły biznesowe, zapewniają, że tylko poprawne informacje przechowywane są w bazie danych.
- Aby ominąć normalne znaczenie znaku w łańcuchu, należy poprzedzić go znakiem backslash (\).
- Funkcja `die` powoduje zatrzymanie skryptu. Opcjonalny argument tej funkcji jest łańcuchem, który będzie przekazany na standardowe wyjście skryptu.

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

```

1 <!DOCTYPE html>
2
3 <!-- Fig. 19.14: form.php -->
4 <!-- Process information sent from form.html, -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Form Validation</title>
9     <style type = "text/css">
10       p { margin: 0px; }
11       .error { color: red; }
12       p.head { font-weight: bold; margin-top: 10px; }
13     </style>
14   </head>
15   <body>
16     <?php
17       // determine whether phone number is valid and print
18       // an error message if not
19       if (!preg_match("/^(\d{3})\d{3}-\d{4}$/", $_POST["phone"]))
20       {

```

Gwarantuje, że numer telefonu ma odpowiedni format.

Rys. 19.14 | Przetwarzanie informacji wysłanych przez form.html. (Część 1/3)

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

```

22   printf( "<p class = 'error'>Invalid phone number</p>
23   <p>A valid phone number must be in the form
24   (555) 555-5555</p><p>Click the Back button,
25   enter a valid phone number and resubmit.</p>
26   <p>Thank You.</p></body></html>" );
27   die(); // terminate script execution
28
29 ?<!-- end PHP script -->
30 <?php print( $_POST["fname"] ); ?>. Thank you for
31 completing the survey. You have been added to the
32 <?php print( $_POST["book"] ); ?>mailing list.</p>
33 <p class = "head">The following information has been saved
34 in our database:</p>
35 <p>Name: <?php print( $_POST["fname"] );
36   print( $_POST["lname"] ); ?></p>
37 <p>Email: <?php print( "Semaj" ); ?></p>
38 <p>Phone: <?php print( "555-5555" ); ?></p>
39 <p>OS: <?php print( $_POST["os"] ); ?></p>
40 <p class = "head">This is only a sample form.
41   you have not been added to a mailing list.</p>
42
43 </body>
44 </html>

```

Przerwa działania skryptu i pozwala na prawidłowe zamknięcie dokumentu HTML.

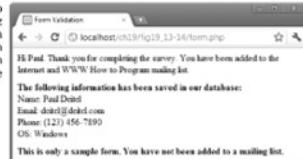
Rys. 19.14 | Przetwarzanie informacji wysłanych przez form.html. (Część 2/3)

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

a) Wysłanie formularza z Rys. 19.13  
przekierowuje użytkownika do form.php, gdzie podawane są odpowiednie instrukcje, jeśli numer telefonu jest w niepoprawnym formacie



b) Wynik form.php po wysłaniu przez użytkownika formularza z Rys. 19.13 z numerem telefonu w poprawnym formacie



Rys. 19.14 | Przetwarzanie informacji wysłanych przez form.html. (Część 3/3)

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

### Obserwacja Inżynierii Oprogramowania 19.1

Aby mieć pewność, że do bazy danych nie trafiają nieprawidłowe dane, należy używać *logiki biznesowej*. Kiedy to możliwe należy sprawdzać poprawność danych na serwerze, ponieważ walidujące skrypty JavaScript mogą być zablokowane przez klienta. Niektóre dane, jak hasła, muszą być zawsze sprawdzane po stronie serwera.

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

### Wskazówka dotycząca zapobiegania błędom 19.3

Należy mieć pewność, że zostaną zamknięte wszystkie otwarte znaczniki HTML5, gdy wywoływana jest funkcja `die`. Niezamknięcie znaczników HTML5 może spowodować, że strona nie będzie wyświetlać się poprawnie w przeglądarce klienta. Funkcja `die` ma opcjonalny parametr, który określa wiadomość na wyjściu podczas końca skryptu. Jedną z technik zamykania znaczników może być wykorzystanie `die("</body></html>")`.

©1992-2012 by Pearson Education, Inc. All Rights Reserved.