

Programowanie aplikacji internetowych

PHP

Statyczne i dynamiczne strony WWW

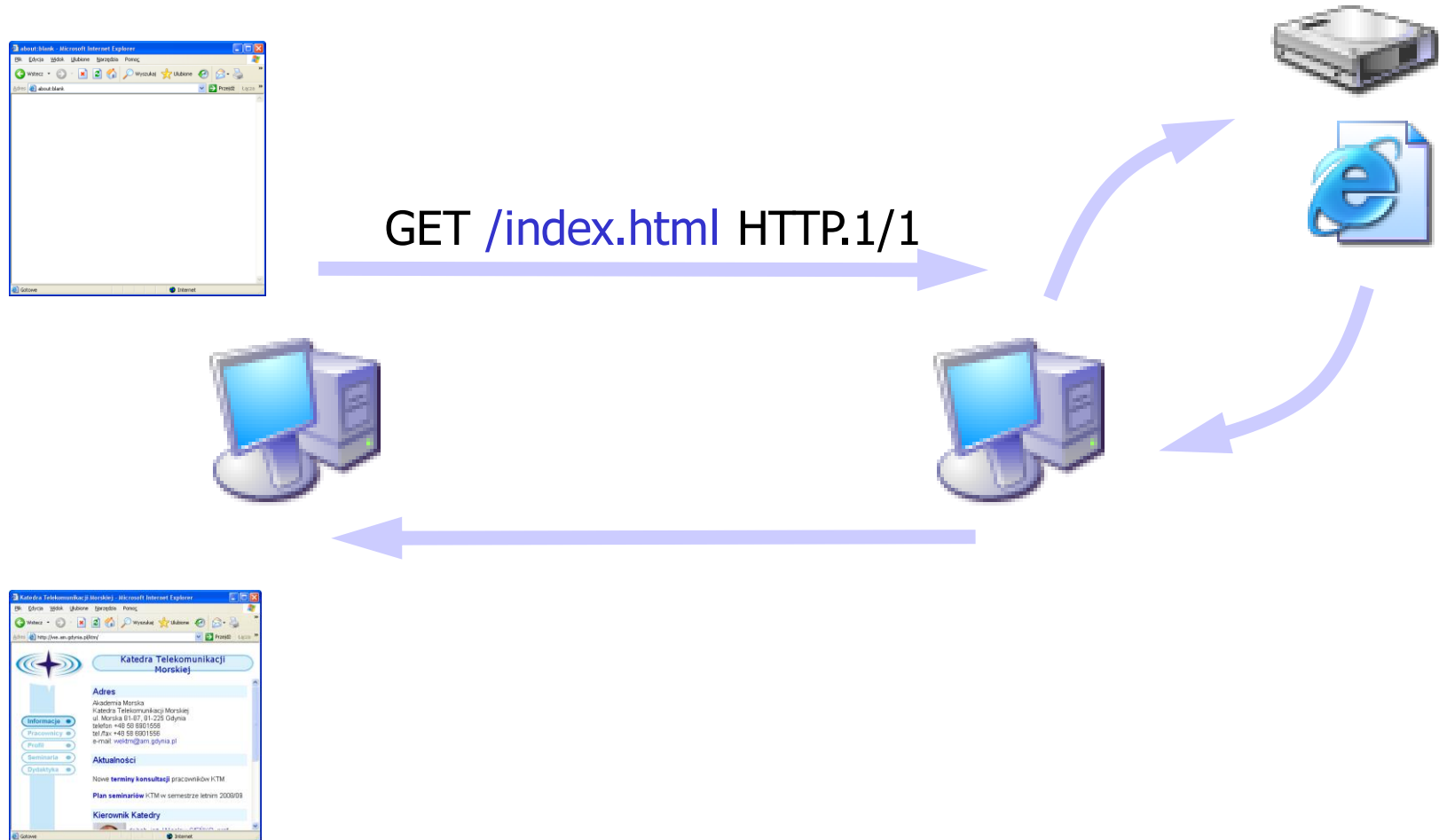
Statyczne strony WWW

- Dokumenty HTML umieszczone na serwerze; zmiana zawartości witryny WWW wymaga modyfikacji plików HTML; niewielkie obciążenie serwera

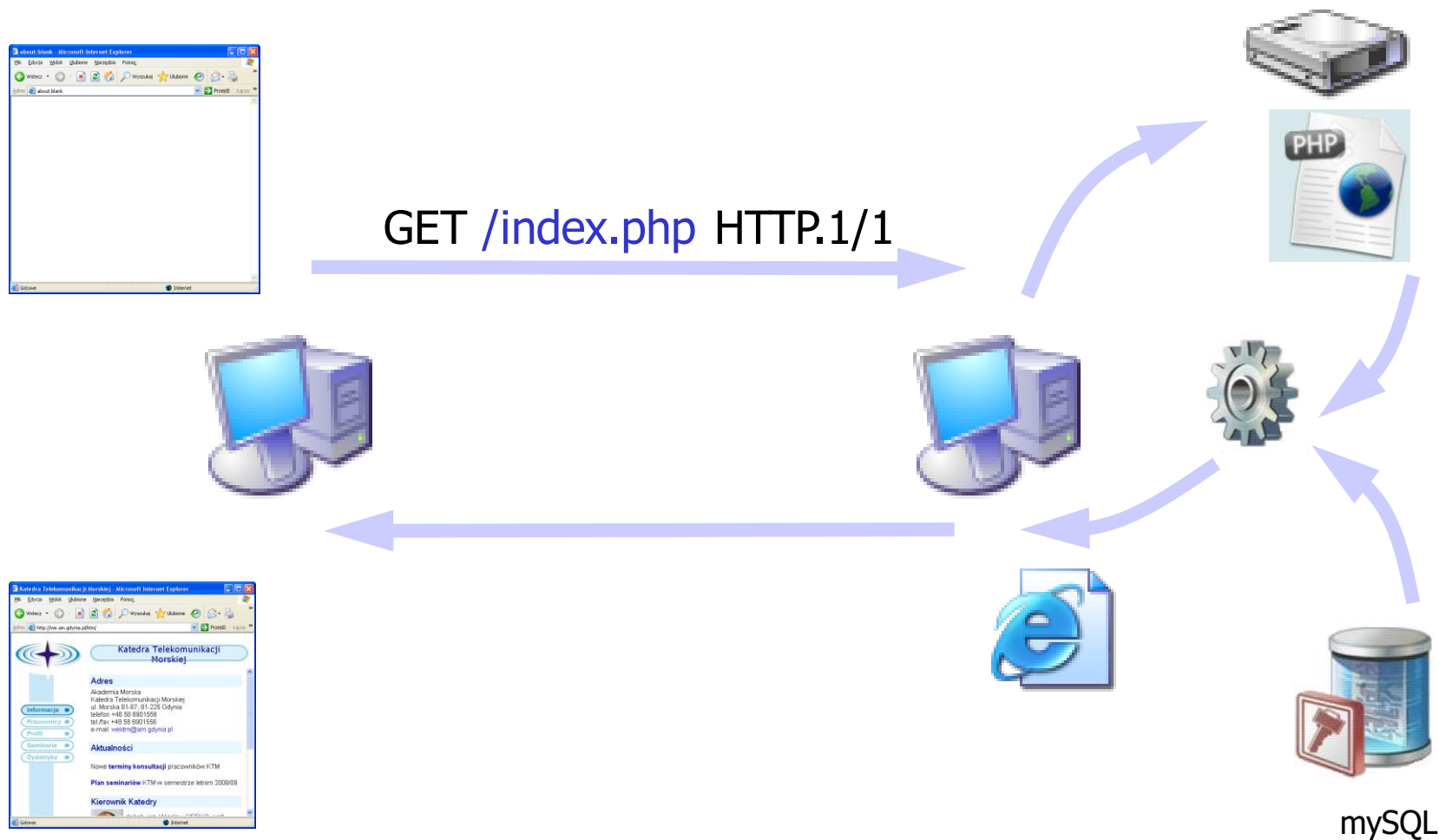
Dynamiczne strony WWW

- Serwer ma dodatkowe moduły (np. interpreter PHP); Zamiast plików HTML na serwerze umieszczone są skrypty generujące dokumenty HTML (np. zamiast x.html → x.php)
- Zawartość dokumentu HTML zwracanego do przeglądarki może być uzależniona od wielu czynników, np. parametrów wybranych przez użytkownika w oknie przeglądarki, zawartości baz danych, odpowiedzi innych serwerów na zapytania itp.
- Większość serwisów do składowania informacji prezentowanych na witrynie wykorzystuje bazy danych

Statyczne strony WWW



Dynamiczne strony WWW



Dynamiczne strony WWW

Zastosowania:

- Portale informacyjne
zawartość informacyjna modyfikowana odrębnymi narzędziami,
zapisywana w bazie danych,
- Sklepy internetowe
dostępne towary opisane w bazie danych, podobnie zamówienie
złożone przez użytkownika,
- Bankowość internetowa
- Systemy nauczania na odległość
materiały dydaktyczne, testy zaliczeniowe, dane instruktorów
oraz słuchaczy (w tym wyniki testów) zapisane w bazie danych
- Fora internetowe, serwisy społecznościowe, serwisy Wiki, ...
- Systemy CMS (*Content Management System*)

Dynamiczne strony WWW

Wymagania:

- Obsługa protokołu HTTP
odczyt i tworzenie nagłówków, obsługa formularzy HTML
- Obsługa baz danych
wbudowane lub dołączane moduły, przynajmniej dla najpopularniejszych systemów baz danych
- Programowanie obiektowe
najbardziej popularna metodyka programowania
- Bezpieczeństwo
obsługa plików cookie (ciasteczek), obsługa sesji,
zabezpieczenia przeciw atakom (np. SQL injection, XSS, ...)

Dynamiczne strony WWW

Platformy:

- Dedykowane serwery
Programy obsługujące komunikację TCP/IP na porcie 80 TCP, dowolny język (np. BCB, Delphi lub MS VS + pakiet InDy)
- Skrypty CGI (*Common Gateway Interface*)
skrypty (Perl, Python) lub programy (dowolny język)
- PHP
skrypty generujące HTML, z możliwością załączania HTML; wbudowane wsparcie HTTP i baz danych
- JSP (*Java Server Pages*), JSF (*Java Server Faces*)
znaczniki (nie HTML), przetwarzane przez JSP/JSF na HTML, środowisko RAD, automatyczne powiązania z kodem (zdarzenia)
- ASP.NET (*Active Server Pages*)
technologia MS, funkcjonalnie podobna do JSF; różne języki programowania (Visual Basic, C#, Java)

Statyczne strony WWW

Dokumenty HTML umieszczone na serwerze

Konfiguracja serwera www określa folder, którego zawartość jest udostępniona przez http/www (żadne pliki spoza tego folderu nie mogą być przez serwer udostępnione), np.:

`/var/www/`

Lokalizacja i nazwa pliku wynika wprost z żądania http (wyjątek – jeżeli w żądaniu jest nazwa folderu, serwer szuka w tym folderze pliku o nazwie domyślnej, zwykle index.html):

GET /	-	/var/www/index.html
GET /style.css	-	/var/www/style.css
GET /img/chopin.jpg	-	/var/www/img/chopin.jpg

Statyczne strony WWW

Dokumenty HTML umieszczone na serwerze

...

Oprócz samego żądania http, serwer uwzględnia również nagłówki http (np. nagłówek "Host: domena.com" – jeżeli na serwerze jest kilka domen)

Nagłówki odpowiedzi generuje serwer

Dynamiczne strony WWW - PHP

Skrypty php umieszczone na serwerze

Przykład skryptu PHP:

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <?php

        echo ("<h1>Hello world!</h1>");

    ?>
  </body>
</html>
```

Dynamiczne strony WWW - PHP

Skrypty php umieszczone na serwerze

Plik php zawiera html i instrukcje php

– fragmenty html oraz dane "drukowane" przez php na standardowe wyjście – funkcja `echo()` – są wysyłane do przeglądarki

Przeglądarka otrzymuje dokument html jak przy stronach www statycznych – nie potrzebuje nic dedykowanego dla php, żeby wyświetlać strony generowane w php

Dynamiczne strony WWW - PHP

Skrypty php umieszczone na serwerze

Pliki php są jednocześnie plikami html, wszystkie zasady odnajdowania i udostępniania plików dotyczą też skryptów php

Lokalizacja i nazwa pliku wynika wprost z żądania http (wyjątek – jeżeli w żądaniu jest nazwa folderu, serwer szuka w tym folderze pliku o nazwie domyślnej, zwykle index.php):

GET /	- /var/www/index.php
GET /style.css	- /var/www/style.css
GET /cars/x.php	- /var/www/cars/x.php
GET /img/logo.jpg	- /var/www/img/logo.jpg

Pliki skryptów php oraz wszelkie inne pliki potrzebne w serwisie (arkusze CSS, obrazki, czcionki, ...) koegzystują w tym samym folderze – pliki "statyczne" są wysyłane do przeglądarki, skrypty – uruchamiane przez interpreter PHP

Dynamiczne strony WWW - PHP

Skrypty php umieszczone na serwerze

...

Uruchamiany jest skrypt otrzymuje wszystkie informacje z żądania http – w tym żądany zasób oraz wszystkie nagłówki; Zasób z żądania może zawierać dane z formularza html, nagłówki mogą zawierać dane z formularza oraz ciasteczka, np.:

```
GET /find.php?str=xxx&options=strict
```

Interpreter udostępnia wszelkie dane z żądania http (nagłówki, dane formularzy, ciasteczka, ...) w postaci tablic, bardzo wygodnej dla programisty, np. `$_GET`, `$_COOKIES`
Do nagłówków odpowiedzi serwera są dodawane nagłówki wygenerowane przez skrypt php, np. nowe wartości ciasteczek

Dynamiczne strony WWW - PHP

Skrypty php umieszczone na serwerze

...

Aplikacje internetowe php, np. systemy CMS, zmieniają domyślny sposób działania serwera przez przekierowania;
W serwerze Apache służy do tego moduł mod_rewrite:

```
--- .htaccess ---  
RewriteEngine On  
RewriteCond %{REQUEST_FILENAME} -f [OR]  
RewriteRule ^ - [L]  
RewriteRule ^(*)$ /index.php?rq=$1 [QSA,NC,L]
```

Niezależnie co będzie w żądaniu, wykonany zostanie skrypt index.php (dane z żądania zostaną przekazane do tego skryptu)

```
GET /studenci -> /index.php?rq=studenci
```

Język PHP

Historia PHP

- 1995: PHP 1.0 – zbiór skryptów języka Perl, później C, pn. PHP (*Personal Home Pages*), autor: Rasmusa Lerdorfa
- 1997: PHP 2.0
- 1998: PHP 3.0 – przepisane od nowa przy wsparciu społeczności użytkowników, autorzy: Z. Suraski, A. Gutmans
- 2000: PHP 4.0 – nowa architektura oraz silnik (*Zend Engine*) – modułowość, wydajność, programowanie obiektowe
- 2004: PHP 5.0 – nowy model programowania obiektowego, obsługa dokumentów XML i nowy silnik obsługi baz danych
- 2005: PHP 5.1 – PDO (*PHP Data Object*) ujednolicony moduł obsługi baz danych (MySQL, MSSQL, Oracle, PostgreSQL)

Język PHP

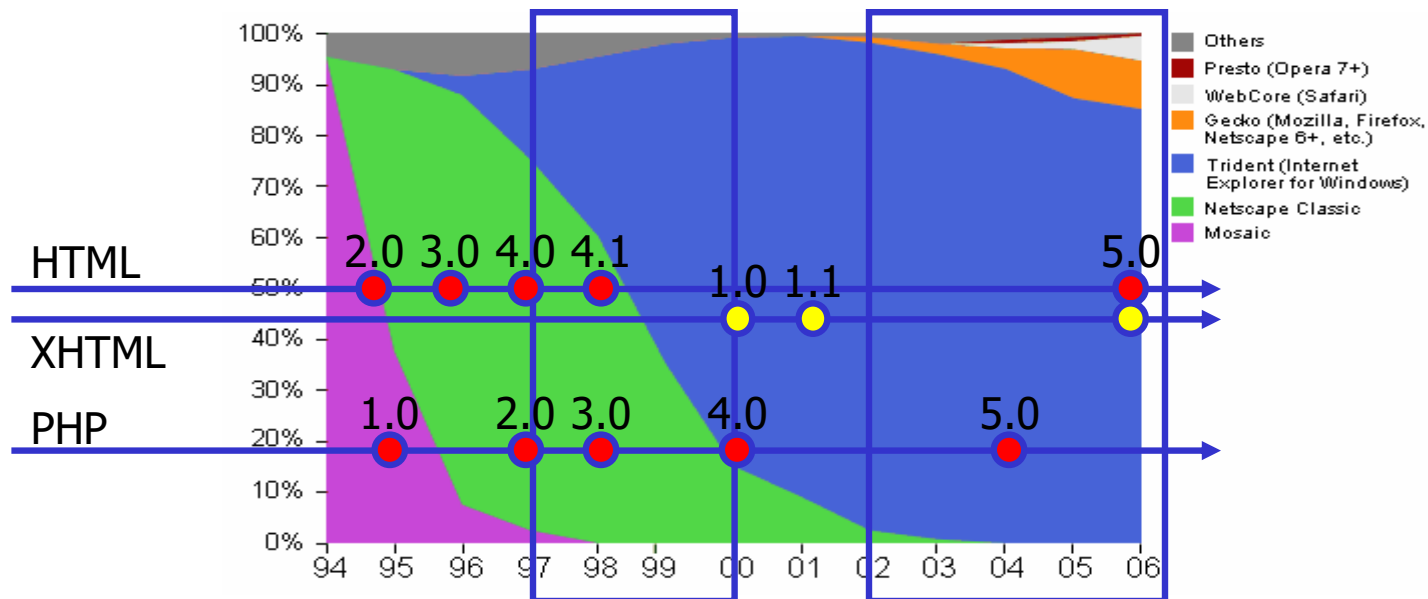
Historia PHP

- ...
- 2005: PHP 6.0 – dodanie obsługi Unicode; Wersja zarzucona
- 2014: PHP 7.0 – zwiększenie wydajności, dopracowanie składni, nowe operatory (?? Oraz $\leq$), typowanie argumentów funkcji; Główny cel – przygotowanie do nowego modelu uruchamiania skryptów (kompilacja JIT, *Just-in-time*)
- 2020: PHP 8.0 – kompilacja *Just-in-time* (poprawa wydajności), większy nacisk na formalną poprawność kodu,

Przeglądarki internetowe

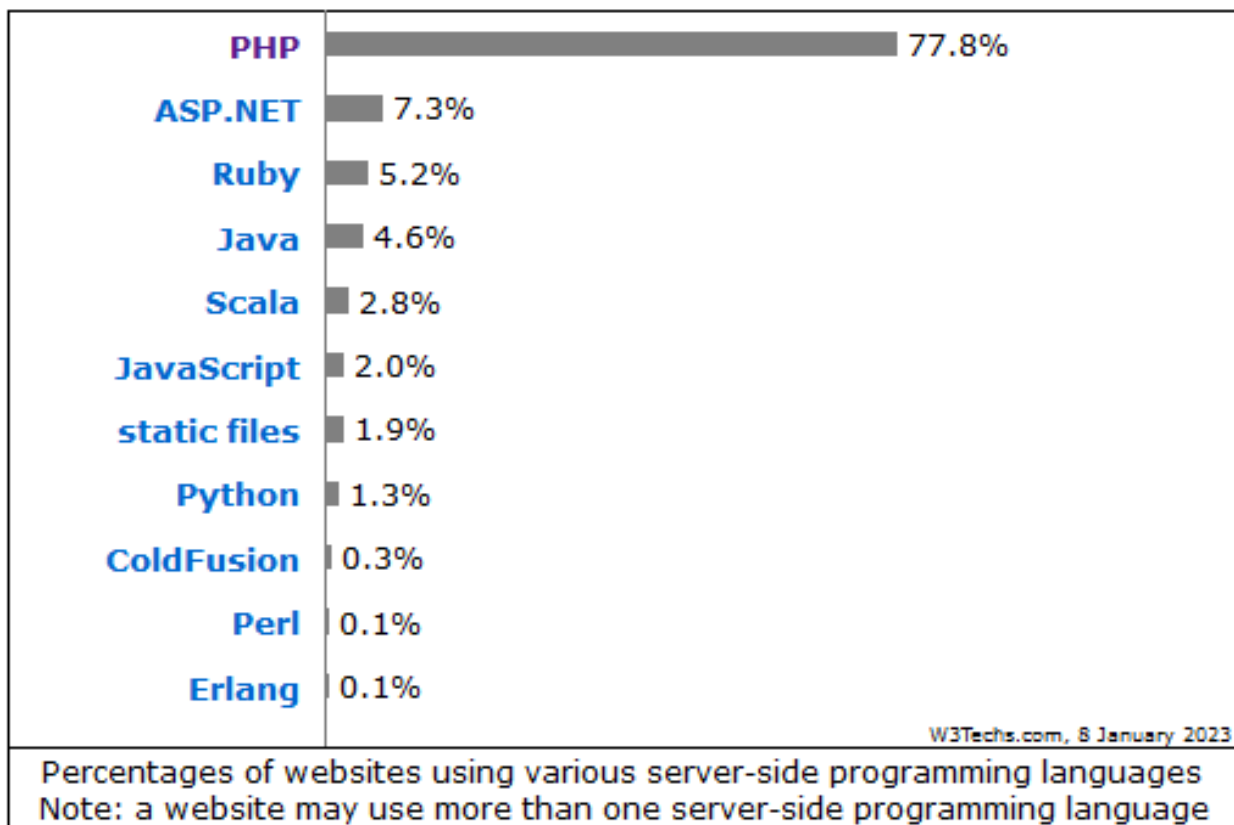
Wojny przeglądarek

- Pierwsza wojna przeglądarek: 1997-1999
MS Internet Explorer vs Netscape Navigator
Niestandardowe rozszerzenia ("fajniejsze" strony WWW);
- Druga wojna przeglądarek: 2002-?
MS Internet Explorer vs (Firefox + Safari + Opera)
Zgodność ze standardami i likwidacja monopoli;



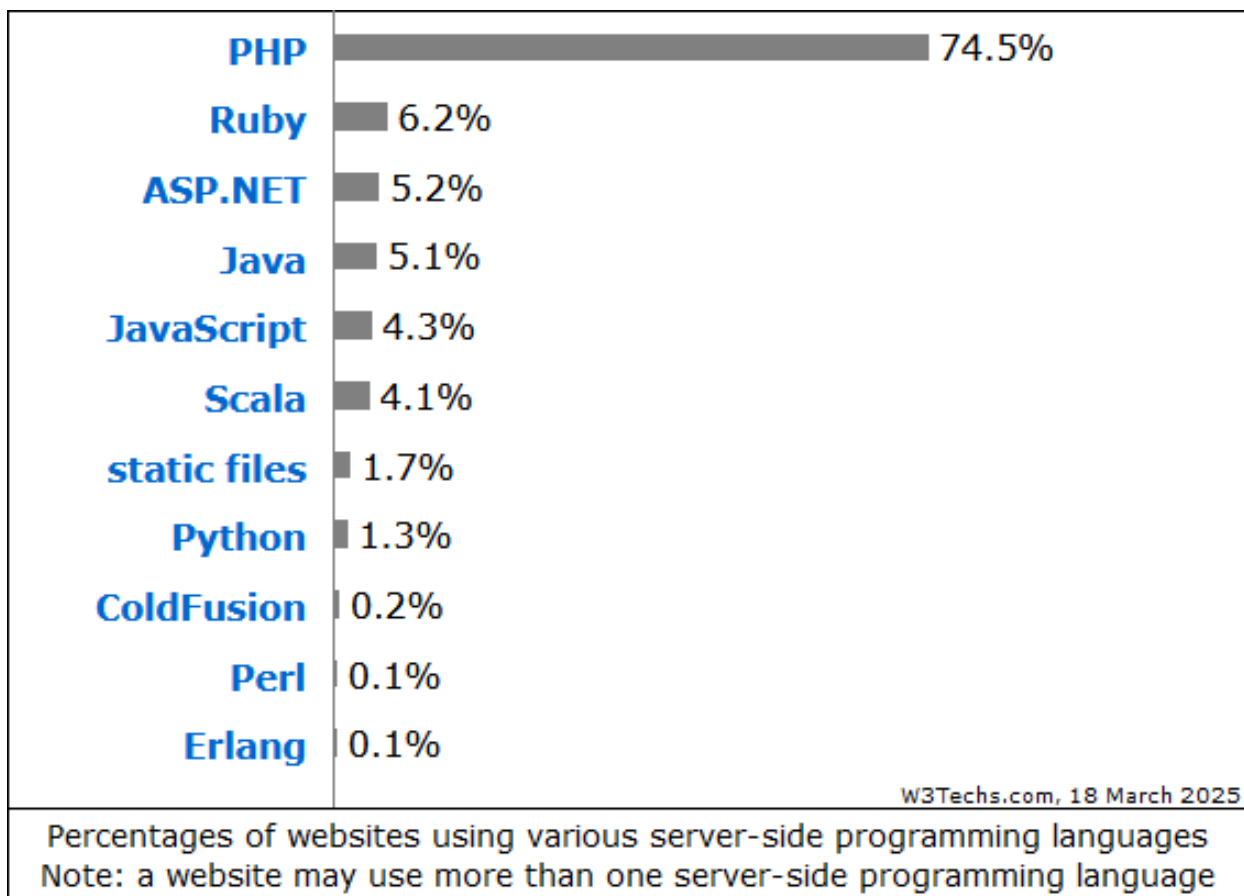
Dynamiczne strony WWW

Języki programowania



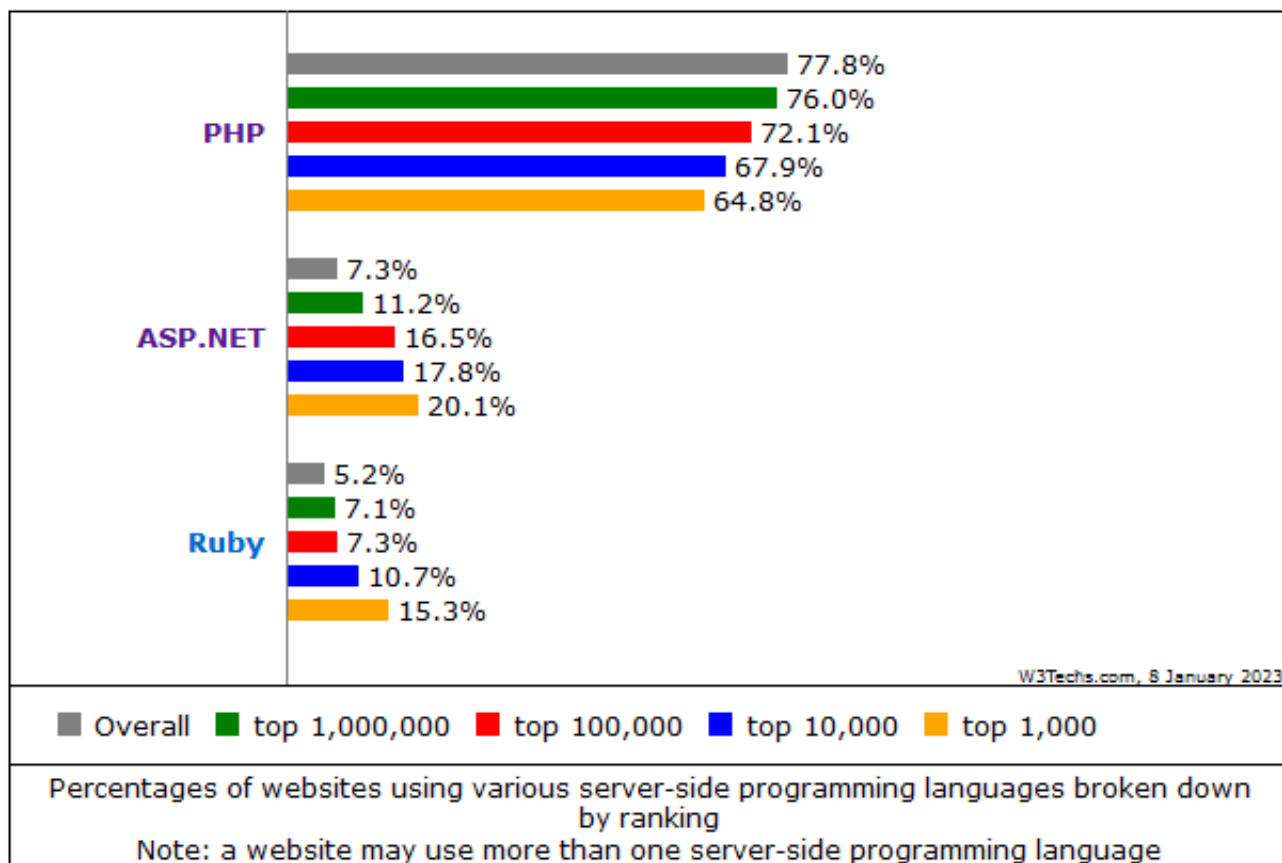
Dynamiczne strony WWW

Języki programowania



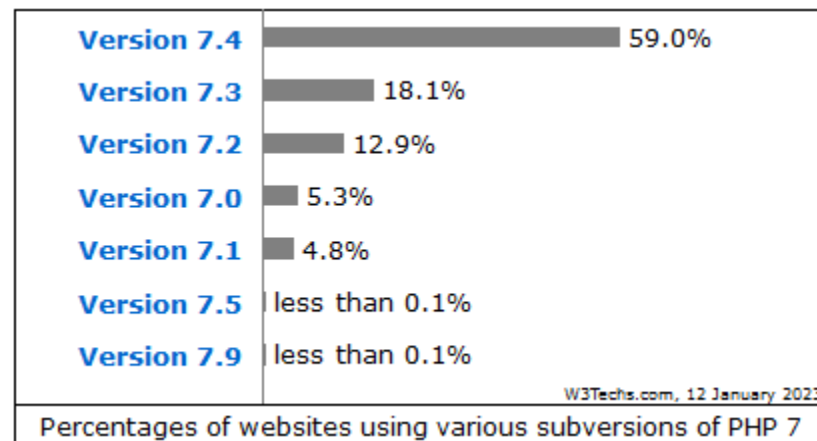
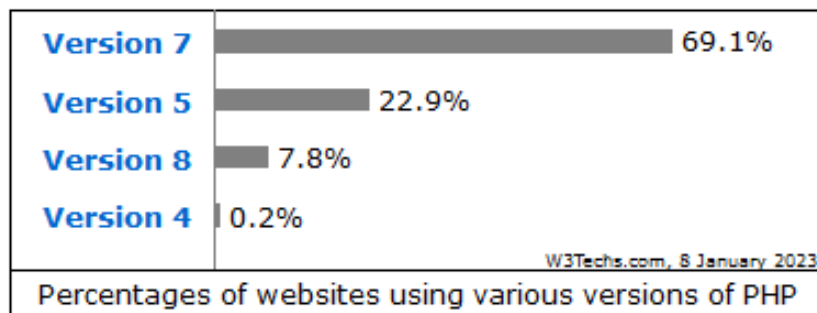
Dynamiczne strony WWW

Języki programowania



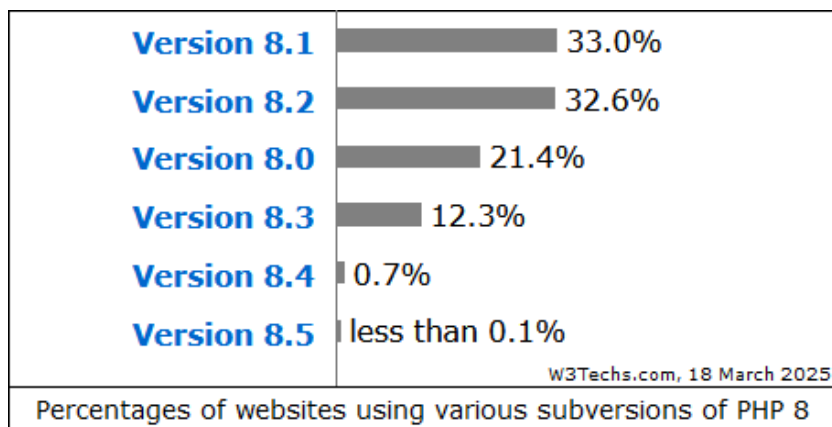
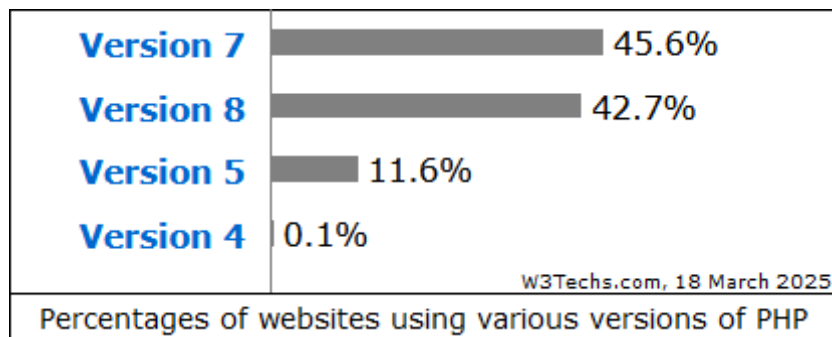
Dynamiczne strony WWW

Większość serwisów używa PHP 7, a dokładniej 7.4



Dynamiczne strony WWW

Wersja 8 coraz częściej zastępuje wersję 7



Język PHP

PHP (*PHP: Hypertext Preprocessor*) język programowania stron WWW; najważniejsze cechy PHP:

- Język skryptowy – programy PHP nie są kompilowane do kodu binarnego, lecz interpretowane na bieżąco podczas wykonywania
- Działa po stronie serwera – skrypt PHP generuje dokument HTML, wyświetlany przez przeglądarkę – nie wymaga określonej przeglądarki, jej wersji lub wtyczek
- Język obiektowy, z potężnymi bibliotekami (w tym PEAR) oraz obsługą baz danych i dokumentów XML
- Opracowany specjalnie do tworzenia dynamicznych stron WWW – wbudowana obsługa HTTP (w tym odczyt i generowanie nagłówek, przekazywanie parametrów dla metod GET i POST, obsługa plików Cookie i sesji, ...), możliwość osadzania HTML

Język PHP

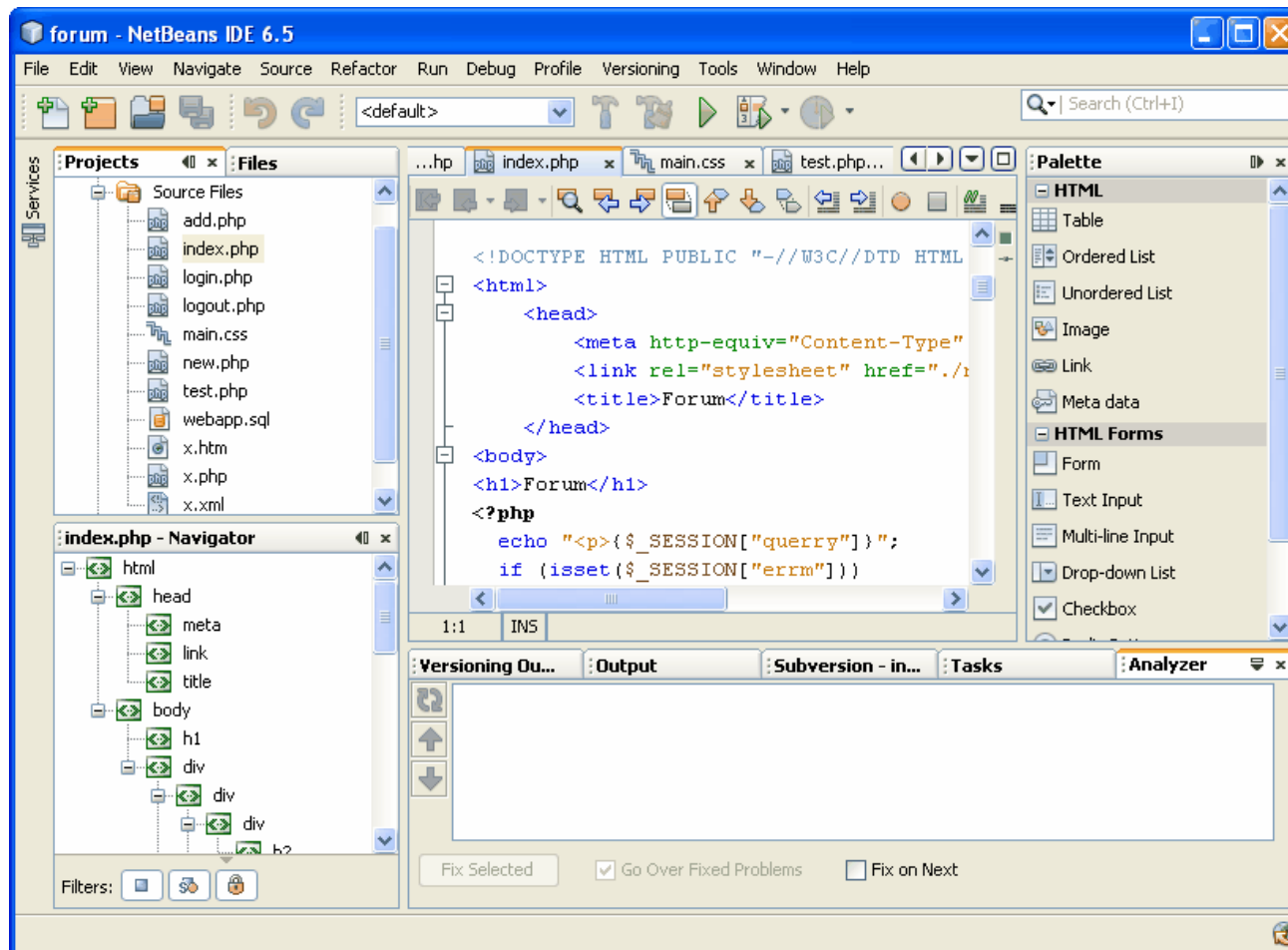
Środowiska programistyczne

- Zend Studio for Eclipse
wtyczka do Eclipse z obsługą PHP
- NetBeans
wersja ze obsługą PHP (w tym Intelli-Sense, składnia, ...),
dostępny też debugger PHP
- VS Code (Microsoft)
- PhpStorm (JetBrains)

- Dowolny edytor tekstu
– język PHP nie wymaga kompilacji!

Język PHP

NetBeans



Języki programowania wysokiego poziomu

Podstawy PHP

Struktura PHP

W pliku PHP można dowolnie mieszać HTML oraz PHP:

- Elementy HTML są kopiowane do wynikowego dokumentu, bez wnikania w sens i treść,
- Instrukcje PHP są wykonywane, a jeżeli ich skutkiem jest zapis w strumieniu wyjścia, trafia to do dokumentu wynikowego
- Przed umieszczeniem w strumieniu wyjścia pierwszego znaku pochodzącego ze skryptu PHP (albo osadzonego w nim HTML), interpreter PHP umieszcza w strumieniu wyjścia potrzebne nagłówki HTTP

Podręcznik PHP

Głównie opis funkcji, ale także opis samego języka, objaśnienie niektórych głównych cech PHP itp.:

<http://pl.php.net/manual/pl/>

Można też znaleźć wiele poradników, przykładów kodu, forów dyskusyjnych itp.

Struktura PHP

Znaczniki PHP – oddzielają instrukcje PHP od HTML osadzonego w skrypcie:

- Element script HTML

```
<script language="php">  
</script>
```

- Znaczniki normalne (tzw. długie)

```
<?php ... ?>
```

- Znaczniki krótkie:

```
<? ... ?>
```

- Znaczniki krótkie w stylu ASP:

```
<% ... %>
```

Wszystko poza znacznikami, jest uznawane za osadzony HTML;
Stosowanie znaczników krótkich wymaga odp. konfiguracji PHP.

Struktura PHP

Przykład skryptu PHP:

- Skrypt "Hello world!":

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <?php

        echo ("<h1>Hello world!</h1>");

    ?>
  </body>
</html>
```

Komentarze w PHP

Dwa rodzaje komentarzy:

- Komentarze liniowe (do końca bieżącej linii):

```
// komentarz
```

```
# komentarz
```

- Komentarze blokowe:

```
/*
```

```
    komentarz
```

```
*/
```

Komentarzy blokowych nie można zagnieżdżać

Zmienne w PHP

PHP jest językiem skryptowym (tj. interpretowanym), stąd sposób traktowania zmiennych jest zupełnie inny, niż w językach kompilowanych. Ogólne zasady:

- Nazwy wszystkich zmiennych zaczynają się od znaku \$
W nazwach można używać liter, cyfr i znaku podkreślenia (dopuszcza się niektóre znaki rozszerzone)
- Zmiennych nie deklaruje się – zmienna jest tworzona, kiedy przypisuje się jej wartość
- Każda zmienna ma typ, nadawany odpowiednio do wartości
- Typ zmiennej może się z czasem zmienić (przy następnym przypisaniu)
- Obowiązuje ścisła kontrola typów (np. podczas wywoływania funkcji, instrukcji przypisania)
- Zmienne typów prostych są automatycznie konwertowane na inne typy proste w razie potrzeby – dotyczy również łańcuchów

Zmienne w PHP

Typy zmiennych prostych w PHP:

- Liczby całkowite
(odpowiednik int C++)

```
$a = 123;  
$b = 0173; // notacja ósemkowa = 123(10)  
$c = 0x7b; // notacja szesnastkowa =  
123(10)
```

- Liczby zmiennoprzecinkowe
(odpowiednik double C++)

```
$x = 12.3;  
$y = 1.23e1;  
$z = 1000000000000000000;
```

Zmienne w PHP

Typy zmiennych prostych w PHP:

- Wartości logiczne
(odpowiednik bool C++)

```
$p = TRUE;    // albo YES  
$q = FALSE;   // albo NO
```

- Typ NULL – "brak" zmiennej
(to nie jest odpowiednik wskazania pustego C++!)

```
$a = 7;  
unset ($a);  
if ($a==NULL)  
    echo "Zmienna \"$a\" nie istnieje";
```

można też wykorzystać funkcję `is_null()`:

```
if (is_null($value)) ...
```

Zmienne w PHP

Typy zmiennych prostych w PHP:

- Łańcuchy znaków – ograniczone cudzysłowami:

```
$s1 = "Hello";  
$s2 = "Znak \" (cudzysłów)";  
$s3 = "<img src='URL'>";
```

Sekwencje ucieczki stosuje się dla znaków:

```
\", \\, \n, \r, \t, \$, \0n, \0xn
```

- Łańcuchy znaków – ograniczone apostrofami:

```
$s4 = 'Hello';  
$s5 = 'Znak \' (apostrof)';  
$s6 = '';
```

Sekwencje ucieczki stosuje się dla znaków:

```
\', \\\
```

Zmienne w PHP

Istnieje istotna różnica w traktowaniu łańcuchów ograniczone apostrofami i cudzysłowami:

- Łańcuchy ograniczone cudzysłowami – interpreter wyłapuje nazwy zmiennych i zastępuje ich wartościami:

```
$s1 = "./images/reddot.png";  
echo "<img src=' $s1 '>";  
echo "<img src=\" $s1 \">>";
```

Można też stosować tzw. notację złożoną:

```
echo "<img src={ $s1 }>";
```

Notacja taka może się przydać również gdy tuż za nazwą zmiennej nie występuje odstęp (np. spacja lub):

```
$s2 = "reddot";  
echo "<img src='./images/{ $s2 }.png'>";
```

Zmienne w PHP

Istnieje istotna różnica w traktowaniu łańcuchów ograniczone apostrofami i cudzysłowami:

- Łańcuchy ograniczone apostrofami
 - nie da się włączyć wartości zmiennej:

```
$s = 100;  
echo 'Val: {$x}';           // Val: {$x}
```

Teoretycznie można stosować specjalny krótki znacznik, ale wymaga to modyfikacji ustawień PHP:

```
echo 'Val: <?= $x ?>';    // Val: 100
```

Zmienne w PHP

Tworzenie HTML przez funkcję echo PHP ma wadę: nie widać kolorowanie składni HTML. Stąd sposób:

- Zamiast włączać HTML do PHP ...

```
<?php
    $s1 = "./images/reddot.png";
    echo "<img src=' $s1 '>";
?>
```

- ... włączamy PHP do HTML:

```

```

Teoretycznie można stosować specjalny krótki znacznik, ale wymaga to modyfikacji ustawień PHP:

```

```

Konwersja typów

Są dwie odmiany konwersji:

- Konwersja automatyczna
zachodzi pomiędzy wszystkimi typami prostymi i jest wykonywana kiedy zajdzie taka potrzeba (np. operacje arytmetyczne):

```
$a = 7;  
$b = "93";  
$c = $a + $b; // 100, o dziwo!  
echo "Wynik: $c";
```

- Konwersja wymuszona
wykonywana jawnie – wymaga zachodu, ale zwiększa kontrolę nad wykonywaniem skryptu:

```
(int) $x          // także (integer) $x  
(double) $x       // także (float) $x  
(string) $x  
(bool) $x         // także (boolean) $x
```

Konwersja typów

Konwersja wymuszona nigdy nie kończy się błędem:

- Łańcuchy nie-liczbowe oraz NULL dają wartość 0:

```
$a = (int) "Seven";    // $a = 0
$b = (int) NULL;       // $b = 0
$c = (int) 1.23e24;    // $c = ???
```

- Wartość 0, 0.0, NULL ciąg pusty oraz "0" dają FALSE, pozostałe wartości i ciagi – TRUE:

```
$p = (bool) $"Trzy"    // TRUE
$q = (bool) $"FALSE"   // TRUE
```

- Konwersja na liczby całkowite polega na odrzuceniu części ułamkowej:

```
$a = (int) 1.9999;      // $a = 1
$b = (int) -2.12e1;     // $b = -21
```

Stałe

Typ i wartość stałych jest określana przy definiowaniu i potem nie może być zmieniona:

- Definicja stałej

```
define (PORT, 80);  
define (DB_USER, "webApp");  
define (DB_PASS, "kernel");
```

- Wykorzystanie stałych zwiększa czytelność kodu i ułatwia pielęgnację

```
$conn = dbManager.connect (DB_USER,  
DB_PASS);
```

Stałą DB_USER można wykorzystać dowolnie wiele razy, ale zmienić nazwę konta bazy danych – wystarczy raz;

Zapisywanie nazw stałych wielkimi literami nie jest obowiązkowe, ale często stosowane – zwiększa czytelność.

Operatory i wyrażenia PHP

Operatory PHP można podzielić na kilka grup:

- Inkrementacji
- Arytmetyczne
- Relacji (porównania)
- Logiczne
- Bitowe
- Znakowe
- Przypisania

Większość operatorów działa jak w C++, jednak są ważne wyjątki. Niektóre operatory mają alternatywny sposób zapisu (wzięty np. z Pascala, jak "and" zamiast "&&")

Operatory i wyrażenia PHP

Operatory arytmetyczne

- + dodawanie
- odejmowanie
- * mnożenie
- / dzielenie
- % reszta z dzielenia

Dzielenie jest zawsze zmiennoprzecinkowe. Chcąc uzyskać wynik całkowity, należy dokonać konwersji:

```
$d = 3 / 4;           // $d = 0.75  
$e = (int) $d;        // $e = 0  
$sr = (int) ($suma/$ilosc)
```

Operatory i wyrażenia PHP

Operatory relacji (porównania)

```
==   równe
===  identyczne
!=   nie równe (także <>)
!==  nie identyczne
<    mniejsze
<=   mniejsze lub równe
>    większe
>=   większe lub równe
<=>  "spaceship"
```

Operator identyczności zwraca TRUE jeżeli zarówno wartość, jak i typ operandów jest taki sam:

```
$p = 15 == "15"; // TRUE
$q = 15 === "15"; // FALSE
```

Operatory i wyrażenia PHP

Operatory logiczne

&& koniunkcja (również "and")
|| alternatywa (również "or")
! negacja

Działanie i sposób użycia jak w C++, np.:

```
$p = $a>0 && !($b<$c || $b<$a);
```

Operatory i wyrażenia PHP

Operatory przypisania

= przypisanie

+= rozszerzone przypisanie

(podobnie -=, *=, /=, %=)

Działanie jak w C++;

Przypisanie jest traktowane jak operacja (jak w C++), a nie instrukcja (jak w Pascalu i Delphi) – operacje te można łączyć:

```
$a = $b = $c = 7;
```

```
$a *= $b += $c = 7;
```

pamiętając przy tym, że wiązanie jest prawe.

Operatory i wyrażenia PHP

Operatory znakowe

- . konkatencja

- .= przypisanie rozszerzone o konkatencję

Operator konkatencji umożliwia tworzenie łańcuchów złożonych z tekstu i wartości zmiennych.

```
$k = "Wynik: " + $suma;
```

```
$l = 100;
```

```
$m = 50;
```

```
echo $l . $m;    // 10050, a nie 150
```

Nie ma ryzyka, że zostanie wykonana konwersja i dodawanie, ponieważ operator konkatencji obsługuje wyłącznie łańcuchy; Analogicznie operator dodawania – wyłącznie liczby;

```
$a = "2";
```

```
$b = "3";
```

```
$c = $a + $b;    // 5, a nie 23
```

Operatory i wyrażenia PHP

Operator trójskładnikowy (ternary)

? :

<warunek> ? <w1> : <w2>

Działanie i sposób użycia jak w C++, np.:

```
$p = $a>0 ? 1 : -1;
```

```
$y = $x==0.0 ? 1.0 : sin($x)/$x;
```

```
if ($x==0.0)
```

```
    $y = 1.0;
```

```
else
```

```
    $y = sin($x)/$x;
```

Operatory i wyrażenia PHP

Operatory koalescencji null

??

<wyrażenie> ?? <wartość_gdy_null>

Działanie i sposób użycia:

```
$p = $_POST["name"] ?? "anonymous";
```

```
if (isset($_POST["name"]))  
{  
    $p = $_POST["name"];  
} else  
{  
    $p = "anonymous";  
}
```

Operatory i wyrażenia PHP

Hierarchia i wiązanie operatorów:

```
prawe      []
prawe      ! ++ -- (typ)
lewe * / %
lewe + - .
lewe < <= > >=
lewe == != <> === !==
lewe &&
lewe ||
prawe      = += -= *= /= itd.
lewe and
lewe or
```

Instrukcje strukturalne PHP

Instrukcje strukturalne

- Instrukcja grupująca (blok instrukcji)
- Instrukcja warunkowa, if-else
- Instrukcja wyboru, switch-case
- Instrukcje iteracyjne
 - z licznikiem, for
 - z warunkiem, while oraz do-while
 - do obsługi tabli, foreach

Instrukcje strukturalne PHP

Instrukcja grupująca

- Składnia

```
{  
    // dowolna liczba instrukcji  
}
```

Bloku należy używać, gdy składnia pozwala użyć jednej instrukcji, a potrzeba wykonać więcej niż jedną czynność

Instrukcje strukturalne

Instrukcja warunkowa (1)

- Składnia

```
if (warunek)  
    instrukcja
```

Jeżeli warunek jest spełniony, instrukcja jest wykonywana, w przeciwnym wypadku instrukcja jest pomijana

- Przykłady

```
if ($x==0)  
    echo "x ma wartość zero";
```

Instrukcje strukturalne

Instrukcja warunkowa (2)

- Składnia

```
if (warunek)  
    instrukcja1  
else  
    instrukcja2
```

Jeżeli warunek jest spełniony, wykonywana jest instrukcja1,
w przeciwnym wypadku instrukcja2

Instrukcje strukturalne

Instrukcja warunkowa (2)

- Przykłady

```
if ($delta>=0)
    echo "<p>nie ma pierwiastków
    rzeczywistych";
else
{
    $x1 = ...;
    $x2 = ...;
    echo "<p>pierwiastki: $x1, $x2";
}
```

Instrukcje strukturalne

Instrukcja iteracyjna z licznikiem

- Składnia

```
for (inicjalizacja; warunek;  
      inkrementacja)  
    instrukcja
```

Na początku wykonywana jest inicjalizacja, która zazwyczaj powoduje nadanie wartości początkowej licznikowi. Następnie sprawdzany jest warunek i jeżeli jest on spełniony, wykonywana jest instrukcja i inkrementacja, po czym czynności te (bez inicjalizacji!) są powtarzane. Jeżeli warunek nie jest spełniony, następuje wyjście z pętli.

Instrukcje strukturalne

Instrukcja iteracyjna z licznikiem

- Przykłady

```
for ($i=1; $i<=10; $i++)  
    echo "<p>{$i}";
```

```
for ($i=2.5; $i<=7.5; $i+=0.5)  
    echo "<p>{$i}";
```

```
for ($i=2; $i<=4096; $i*=2)  
    echo "<p>{$i}";
```

Instrukcje strukturalne

Instrukcja iteracyjna z licznikiem

- Przykłady

```
<h2>Tytuł</h2>
```

```
<?php
```

```
for ($i=1; $i<=10; $i++)  
{  
    $k = $i*$i;  
    echo "<p> {$i} -> {$k}";  
}  
?>
```

Instrukcje strukturalne

Instrukcja iteracyjna z licznikiem

- Przykłady

```
<table>
<tr> <th>liczba <th>kwadrat
<?php
for ($i=1; $i<=10; $i++)
{
    $k = $i*$i;
    echo "<tr> <td>{$i} <td>{$k} ";
}
?>
</table>
```

Instrukcje strukturalne

Instrukcja iteracyjna z warunkiem (1)

- Składnia

```
while (warunek)  
    instrukcja
```

Najpierw sprawdzany jest warunek i jeżeli jest on spełniony, wykonywana jest instrukcja, po czym czynności te są powtarzane. Jeżeli warunek nie jest spełniony, następuje wyjście z pętli.

Instrukcje strukturalne

Instrukcja iteracyjna z warunkiem (1)

- Przykłady

```
$i=10;  
while ($i>0)  
{  
    echo "<p>{$i}";  
    $i--;  
}
```

Instrukcje strukturalne

Instrukcja iteracyjna z warunkiem (2)

- Składnia

do

instrukcja

while (*warunek*) ;

Najpierw wykonywana jest instrukcja, a następnie sprawdzany jest warunek. Jeżeli jest on spełniony, to czynności są powtarzane.

Jeżeli warunek nie jest spełniony, następuje wyjście z pętli.

Instrukcje strukturalne

Instrukcja iteracyjna z warunkiem (2)

- Przykłady

```
$d=105;  
$p=14;  
do  
{  
    $r = $d % $p;  
    $d = $p;  
    $p = $r;  
}  
while ($r!=0);  
echo "NWP (105, 14) = $d";
```

Instrukcje strukturalne

Instrukcja iteracyjna do obsługi tablic

- Składnia

```
foreach (tablica as [klucz => ] wartość)  
    instrukcja
```

Instrukcja wykonywana jest kolejno dla wszystkich elementów tablicy (kolejność według pozycji na liście, a nie wartości indeksów)

Instrukcje strukturalne

Instrukcje sterujące wykonywaniem pętli

- Instrukcja `break`
powoduje wyjście z aktualnie wykonywanej pętli

```
if (warunek)  
    break;
```

- Instrukcja `continue`
powoduje przerwanie aktualnie wykonywanej iteracji

```
if (warunek)  
    continue;
```

Instrukcje `break` oraz `continue` są używane zawsze w połączeniu z instrukcją warunkową

Instrukcje strukturalne

Instrukcje sterujące wykonywaniem pętli

- Przykłady

```
for ($i=-5; i<=5; $i++)  
{  
    if ($i==0)  
        continue;  
    $o = 1.0/$i;  
    echo "<p> 1/$i = $o";  
}
```

Instrukcje strukturalne

Instrukcje sterujące wykonywaniem pętli

- Przykłady

```
$d=105;  
$p=14;  
do  
{  
    $r = $d % $p;  
  
    $d = $p;  
    $p = $r;  
}  
while ($r!=0);  
echo "NWP (105, 14) = $d";
```

Instrukcje strukturalne

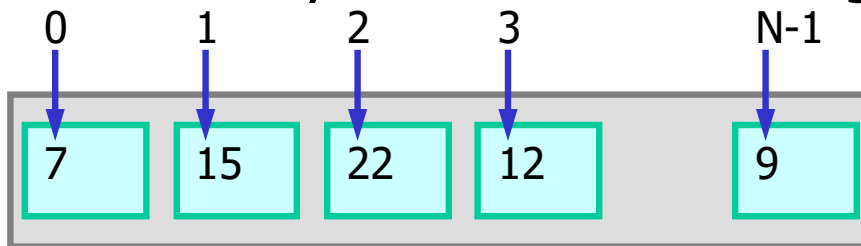
Instrukcje sterujące wykonywaniem pętli

- Przykłady

```
$d=105;  
$p=14;  
do  
{  
    $r = $d % $p;  
    if ($r==0) break;  
    $d = $p;  
    $p = $r;  
}  
while ($r!=0);  
echo "NWP (105, 14) = $p";
```

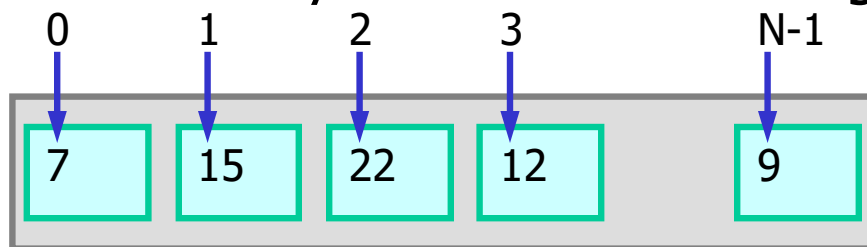
Tablice

W większości języków programowania tablica to indeksowany zbiór elementów tego samego typu:

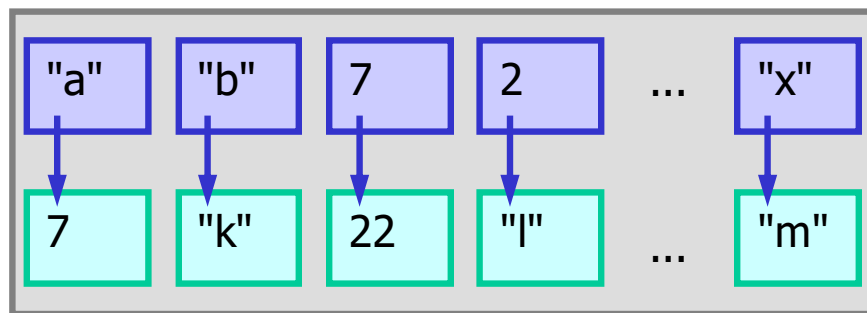


Tablice

W większości języków programowania tablica to indeksowany zbiór elementów tego samego typu:



W języku PHP tablice mają inny charakter: są to tablice asocjacyjne (inaczej odwzorowania), składające się z par klucz-wartość (ang. *key-value*), generalnie dowolnego typu:



Tablice

Tworzenie tablic

- Do tworzenia tablic należy używać funkcji `array()`;
Można utworzyć tablicę pustą lub zainicjować ją wartościami

```
$t1 = array ();
```

```
$t2 = array ("zero", "jeden", "dwa");
```

Domyślnie klucze to liczby naturalne (0, 1, 2,... - jak w C++), jednak można wskazać dowolne klucze, dowolnego typu:

```
$t3 = array (7 => "siedem", 3 => "trzy");
```

```
$t4 = array ("dwa" => "two", "mój" =>  
"my");
```

```
$t5 = array (2 => "dwa", "jeden" => 1);
```

Tablice z indeksami w postaci łańcuchów znaków są powszechnie wykorzystywane (np. zmienne superglobalne `$GLOBALS`, `$_SERVER`, `$_REQUEST`, `$_POST`, `$_GET`), jednak mieszanie różnych typów indeksów nie jest zalecane;

Tablice

Dodawanie i usuwanie elementów tablic

- Do tablicy można dodawać elementy na dwa sposoby,
 - podając indeks:

```
$t = array (7 => "siedem", 3 => "trzy");  
$t[7] = "seven"; // zmiana wartości  
$t[2] = "two";    // dodanie elementu
```

- nie podając indeksu:

```
$t[] = "osiem";
```

nowo utworzony element ma indeks w postaci liczby naturalnej, o jeden większej niż indeks dotąd największy (a nie ostatni)

- Usuwanie elementu wymaga użycia funkcji unset:

```
unset ($t[3]);    // usuwa parę [3]=>"trzy"
```

Tablice

Dostęp do elementów tablic

- Element dostępny jest przez indeks (klucz), jednak w tym wypadku automatyczna konwersja typów nie działa – należy użyć klucza odpowiedniej wartości i typu:

```
$t = array (7 => "siedem", "trzy" => 3);  
echo $t[7];           // "siedem"  
echo $t["trzy"];      // 3  
echo $t[3];           // 0 (element nie  
istnieje!)
```

Jako indeksu (klucza) można też użyć zmiennej lub wyrażenia, należy tylko zadbać o zgodność typu:

```
$i = "trzy";  
echo "$i: {$t[$i]}";
```

Tablice

Dostęp do elementów tablic

- Można przetwarzać zawartość tablic przy pomocy pętli for; jest to łatwiejsze, gdy stosowane są indeksy liczbowe:

```
for ($i=0; $i<10; $i++)  
    $t[$i] = random() % 100; // 1. losowa  
0..99
```

```
for ($i=0; $i<10; $i++)  
    echo "<p>{$t[$i]}";
```

```
sort ($t);  
for ($i=0; $i<10; $i++)  
    echo "<p>{$t[$i]}";
```

Tablice

Dostęp do elementów tablic

- Tablice z indeksami w postaci łańcuchów tekstowych najwygodniej jest przetwarzać przy pomocy pętli foreach. Instrukcja wykonywana jest kolejno dla wszystkich elementów tablicy

```
foreach ($t as $k => $w)  
    echo "<p>klucz: {$k}, wartość: {$w}";
```

Klucz jest elementem opcjonalnym i można go pominąć:

```
foreach ($t as $w)  
    echo "<p>wartość: {$w}";
```

Tablice

Instrukcja iteracyjna do obsługi tablic

- Przykłady

```
<h2>Zawartość tablicy</h2>
```

```
<?php
```

```
foreach ($t as $key => $val)  
    echo "<p> {$key} -- {$val}";  
?>
```

Tablice

Instrukcja iteracyjna do obsługi tablic

- Przykłady

```
<table>
```

```
<tr> <th>klucz <th>wartość
```

```
<?php
```

```
foreach ($t as $key => $val)
```

```
    echo "<tr> <td>{$key} <td>{$val}";
```

```
    ?>
```

```
</table>
```

Przydatne funkcje PHP

Funkcja `php_info()`

- `phpinfo()` – drukuje (do HTML) konfigurację PHP.

Umożliwia to sprawdzenie wielu ustawień, począwszy od drobiazgów (np. krótkie znaczniki PHP), przez dostępność różnych funkcjonalności (np. przesyłanie plików na serwer), aż do ustawień związanych z bezpieczeństwem (np. sposób przesyłania danych o sesji połączenia klient-serwer) .

Przydatne funkcje PHP

Funkcje dotyczące zmiennych

- `var_dump` – drukuje (do HTML) typ i wartość zmiennej

```
$x = 7;  
var_dump($x);      // int 7  
  
$s = "Hello!";  
var_dump($s);      // string(6) "Hello!"  
  
$t = array (3=>"trzy", 2=>"dwa");  
var_dump($t);  
// array(2) { [3]=> string(4) "trzy", ...
```

Przydatne funkcje PHP

Funkcje dotyczące zmiennych

- `isset($x)` – sprawdza, czy wskazana zmienna istnieje
- `unset($x)` – niszczy zmienną

```
$x = 7;
```

```
$e = isset($x);    // TRUE
```

```
unset($x);
```

```
$e = isset($x);    // FALSE
```

Przydatne funkcje PHP

Funkcje drukujące do strumienia HTML

- echo

```
echo string arg1, [string arg2, ...];
```

drukuję do HTML wartości kolejnych argumentów

```
echo "suma: $s";
```

```
echo "suma: ". $s;
```

```
echo "suma: ", $s;
```

```
echo "suma: $t[$i]";
```

```
echo "suma: {$t['trzy']}";
```

```
echo "suma", $t['trzy'];
```

```
echo 'tylko tekst: $x';
```

Przydatne funkcje PHP

Funkcje drukujące do strumienia HTML

- print

```
int print (string arg);
```

drukuję do HTML wartość argumentu

```
print ("suma: $s");  
print "suma: $s";  
print "suma: ". $s;  
print "wynik: {$t['trzy']}";
```

print() działa bardzo podobnie jak echo(); zasadnicza różnica polega na tym , że print akceptuje tylko jeden argument;

Obie funkcje nie są "prawdziwymi" funkcjami – są to specyficzne konstrukcje języka PHP

Przydatne funkcje i klasy PHP

Biblioteki PHP zawierają setki funkcji i klas, m.in.:

- Matematyczne
- Manipulacji łańcuchami znaków, w tym Unicode
- Wyrażenia regularne (regex)
- Obsługi dokumentów XML
- Obsługi daty/czasu
- Obsługi plików (szukanie, przenoszenie, zapis/odczyt)
- Obsługi baz danych (MySQL/MarioDB, PostgreSQL, SQLite, ...)
- Grafiki (tworzenie i konwersja/przekształcanie)
- Usług sieciowych (np. SOAP, cURL, ...)

Biblioteki zewnętrzne, m.in.:

- PHPMailer – poczta elektroniczna
- TCPDF – generowanie dokumentów pdf