**1.PHP jest skryptowanym językiem programowania za pomocą którego można tworzyć aplikacje WWW:**

- generować dynamicznie zawartość strony internetowej

- tworzyć i edytować pliki na serwerze

- odbierać i wysyłać cookies

-wykonywać operacje na bazie danych

- sterować dostępem użytkowników do stron wizyty internetowej

- szyfrować dane

**2. Struktura języka**

Skrypty PHP są plikami tekstowymi, dlatego do ich pisania można wykorzystać dowolny edytor tekstu. Aby pliki tekstowe były prawidłowo rozpoznawane przez serwer WWW muszą mieć rozrzedzenie .php

**3. Blok instrukcji PHP**

Umieszczając skrypt php w dokumencie HTML, należy zaznaczyć w kodzie, gdzie zaczynają się kończą polenienia PHP Do tego celu służą specjalne znaczniki

Standardowy- <?Php ?>

Skrócony <? ?>

**4. Przykłady:**

<?

Echo „Mój pierwszy skrypt PHP”;

?>

<?php

Print („Mój pierwszy skrypt PHP”);

?>

<!DOCTYPE HTML>

<html>

<head>

<title>Skrypt PHP</title>

<meta charset=”UTF-8”>

</head>

<body>

<p>

<?php

Echo „Mój pierwszy skrypt PHP”;

?>

</p>

</body>

</html>

**5. Komentarze**

- komentarz blokowy /\* \*/

-komentarz jednowierszowy //

-komentarz jednowierszowy uniksowy #

**6. Zmienne**

- musi się zaczynać od litery lub znaku pokreślenia

- może składać się jedynie z liter, cyfr i znaku podkreślenia

- w nazwach rozróżniane są małe i duże litery

- w nazwach może stosować polskie litery

- zaczyna się $

**7. Przyklady:**

<?php

$x=1;

$nazwa\_1 = „tekst”;

$liczba7 = 345;

$ilość = 4;

Echo „wynik wynosi $x <br>”;

Echo „To jest $nazwa\_1 <br>”;

?>

**8. zmienne globalne**

- $\_GET – tablica zawierająca zmienne przysyłane do skryptu za pomocą metody GET

- $\_POST – tablica zawierająca zmienne przysyłane de skryptu za pomocą metody POST

- $\_COOKIE – tablica zawierająca zmienne przysyłane de skryptu za pomocą cookies

- $\_FILES – tablica zawierająca zmienne przysyłane do skryptu podczas przesyłania plików

- $\_SERVER

- $\_ENV - tablica zawierająca zmienne zawierające zmienne środowiskowe

- $\_REQUEST - tablica zawierająca zmienne przysyłane do skryptu przez użytkownika

-$\_SESSION – jest to tablica zawierająca zmienne zarejestrowane w bieżącej sesji

-$GLOBALS – jest to tablica zawierająca odniesienie do każdej zmiennej utworzonej przez użytkownika, która ma zasięg globalny dla danego skryptu.

**9.Typy danych**

Boolean- Jest to typ logiczny. Przyjmuje jedną z dwóch wartości Prawda Fałsz

Integer- Jest to typ liczb całkowitych. Może służyć do przedstawiania dodatnich i ujemnych, liczb całkowitych. Liczby mogą zostać zapisane w formacie dziesiętnym, ósemkowym lub szesnastkowym.

Float- Jest to liczba zmienno przecinkowa. Może przedstawiać zarówno dodatnie, jak i ujemne liczby rzeczywiste.

String- Jest to typ łańcuchowy, który służy do zapamiętywania znaków. Pojedynczy znak ciągu jest zapamiętywany w jednym bajcie.

Array – Tablice przechowywują zbiory danych najczęściej jednego typu, a dostęp do nich jest możliwy przez index. Tablice moją być jednowymiarowe lub wielowymiarowe.

Tablice asocjacyjne, czyli tablice skojarzeniowe, to tablice w których zamiast indeksów liczbowych używa się identyfikatorów. W takich tablicach przechowuje się pary danych: unikatowy klucz i wartość. Dostęp do wartości uzyskuje się poprzez podanie wartości klucza. Dla tablic indeksowanych klucz jest liczbą, dla tablic asocjacyjnych jest on ciągiem znaków mających określone znaczenie.

**10. Zmiana typu zmiennej**

W języku PHP podczas przypisywania zmiennej nowej wartości ponownie ustalany jest jej typ. Czasami zachodzi jednak konieczność zmiany typu danych. Można to zrobić za pomocą rzutowania (cast) – w tedy efekt jest jednorazowy – lub za pomocą funkcji settype() – wówczas efekt jest trwały.

Rzutowanie typów odbywa się przez podanie nowego typu w nawiasie przed zmienną lub wartością , której typ należy zmienić dozwolone są określone typy rzutowań, są to:

- integer – rzutowanie do typu całkowitego

- float – rzutowanie dl typu rzeczywistego

- string – rzutowanie do ciągu tekstu

- array – rzutowanie do tablicy

- object – rzutowanie do obiektu

Przykład:

<?php

$x = 23.75;

$y = (integer) $x;

Echo „$x <br>”;

Echo „$y”;

?>

Zmianę typu w sposób trwały przeprowadza się za pomocą funkcji settype(). Funkcja ta ma dwa argumenty. Pierwszym jest nazwa zmiennej, której nadajemy nowy typ, drugi parametr określający nowy typ zmiennej. Parametr ten może przyjmoać wartości: integer, float, string , array, object.

Settype($zmienna, ‘nowy typ’);

Jeżeli zmiana typu przebiegła pomyślnie funkcja zwraca true, jeśli nie false.

Przykład:

<?php

$x = 97.234;

Echo „Zadaklerowana wartość zmiennej \$x: $x <br>”;

Settype($x, ‘string’);

Echo „Wartość zmiennej \$x po zmianie typa na string: $x <br>”;

Settype($x, ‘integer’);

Echo „Wartość zmiennej \$x po zmianie typu na integer: $x <br>”;

?>

**11. Stałe**

Zmienna której nie można zmienić

Deficja stałej ma postać:

Define(„NAZWA\_STAŁEJ”, wartość);

Przykład:

<?php

Define(„WIEK”. „21”);

Echo „mamy wiek „ . WIEK;

?>

**12. Operatory**

arytmetyczne:

|  |  |  |
| --- | --- | --- |
| dodawanie | + | $a + $b |
| odejmowanie | - | $a - $b |
| mnożenie | \* | $a \* $b |
| dzielenie | / | $a / $b |
| Dzielenie modulo | % | $a % $b |
| potęgowanie | \*\* | $a \*\* $b |

Porównania:

|  |  |  |
| --- | --- | --- |
| Wynik true, gdy argumenty są równe | == | $a == $b |
| Wynik true, gdy argumenty są różne | != | $a != $b |
| Wynik true, gdy argumenty są tego samego typu i są równe | === | $a === $b |
| Wynik true, gdy argumenty są różne lub są różnych typów | !== | $a !== $b |
| Wynik true, gdy argument pierwszy jest większy od drugiego | > | $a > $b |
| Wynik true, gdy argument pierwszy jest mniejszy od drugiego | < | $a < $b |
| Wynik true, gdy argument pierwszy jest większy od drugiego lub mu równy | >= | $a >= $b |
| Wynik true, gdy argument pierwszy jest mniejszy od drugiego lub mu równy | <= | $a <= $b |
| Wynik true, gdy argumenty są różne | <> | $a <> $b |

Bitowe:

|  |  |  |
| --- | --- | --- |
| Iloczyn bitowy (AND) | & | $a & $b |
| Suma bitowa (OR) | | | $a | $b |
| Negacja bitowa (NOT) | ~ | $a ~ $b |
| Bitowa różnica symetryczna | ^ | $a ^ $b |
| Przesunięcie bitowe w prawo | >> | $a >> $b |
| Przesunięcie bitowe w prawo | << | $a << $b |

Logiczne:

|  |  |  |
| --- | --- | --- |
| Iloczyn logiczny | and | $a and $b |
| Iloczyn logiczny | && | $a &&$b |
| Suma logiczna | or | $a or $b |
| Suma logiczna | || | $a ||$b |
| Negacja logiczna (NOT) | ! | $a !$b |
| Różnica logiczna | xor | $a xor $b |

Operatory przypisania:

Przykładowo zapis: $a += 4 oznacza to samo co: $a = $a +4

|  |  |  |
| --- | --- | --- |
| $x = 10 | = | $x = 10 |
| $x += 5 | += | $x = $x + 5 |
| $x -= 5 | -= | $x = $x - 5 |
| $x \*= 5 | \*= | $x = $x \*5 |
| $x /= 5 | /=! | $x = $x /5 |
| $x %= 5 | %= | $x = $x % 5 |

Operatory łańcuchowe:

|  |  |  |
| --- | --- | --- |
| Łączenie łańcuchów znakowych | . | $x = „moje „ . „miasto”; |
| Dołączanie do łańcucha znakowego | .= | $x = „moje „;  $x .= „miasto”; |

Przykład:

<?PHP

$osoba["nazwisko"] = "kowalski";

$osoba["imie"] = "Jan";

$osoba["wiek"] = 27;

echo $osoba["nazwisko"] . " " . $osoba["imie"] . " ma " . $osoba["wiek"] . " lat.";

?>

Operatory inkrementacji i dekrementacji:

Operator inkrementacji powoduje zwiększenie wartości o jeden. Może występować w postaci przedrostkowej (++$x) lub przyrostkowej ($x++). Operacja $x++ zwieksza wartość zmiennej po jej wykorzystaniu, natomiast ++$x przed jej wykorzystaniem.

Operator dekrementacji działa analogicznie, tylko zamiast zwiększać wartości zmiennych, zmniejsza je.

Przykład:

<?php

$x = 7;

echo $x++; echo "<br>";

echo ++$x; echo "<br>";

$v = $x;

$t = $x++;

$z = $x;

$y = ++$x;

echo "Wartość zmiennej \$v = $v <br>";

echo "Wartość zmiennej \$t = $t <br>";

echo "Wartość zmiennej \$z = $z <br>";

echo "Wartość zmiennej \$y = $y <br>";

?>

**13. Instrukcje Sterujące**

**If:**

If(warunki){

}

Przykład:

<?php

$x = 11;

$y =7;

If ($y > $x)

Echo „Wartość zmiennej y jest wieksza od wartości zmiennej x.”;

If ($y < $x)

Echo „Wartość zmiennej x jest wieksza od wartości zmiennej y.”;

?>

**If else:**

Jeżeli trzeba sprawdzić dwa wykluczające się warunki, można użyć rozbudowanej instrukcji warunkowej w postaci:

If ( warunek)

Intrukcja;

Else

Instrukcja;

Przykład:

$x = 11;

$y =7;

If ($y > $x)

Echo „Wartość zmiennej y jest większa od wartości zmiennej x.”;

else

Echo „Wartość zmiennej x jest większa od wartości zmiennej y.”;

?>

**Elseif:**

Kolejne instukcje :

If( warunek )

Intrukcja;

Elseif(warunek)

Intrukcja

Elseif(warunek)

Intrukcja

Przykład:

<?php

$X = 11;

$y = 7;

$z = $x + $y;

If ($z < 0)

echo "wartość zmiennej z jest ujemna.";

elseif ($z < 10)

echo "wartość zmiennej jest zawarta w zakresie od 0 do 10";

elseif ($z < 20)

echo "wartość zmiennej jest zawarta w zakresie od 10 do 20";

elseif ($z < 30)

echo "wartość zmiennej jest zawarta w zakresie od 20 do 30";

else

echo "wartość zmiennej z jest większa od 30";

?>

**Switch:**

Działanie instrukcji wygląda następująco: sprawdz wartość wyrażenia i jeżeli wynikiem jest wartość1, wykonaj instrukcje1 i wyjdź z bloku switch (break). Jeśli wynikiem jest wartość2, to wykonaj instrukcje2 i wyjdź z bloku switch. Jeśli wynikiem jest wartość3, to wykonaj instrukcje3 i wyjdź z bloku switch. Jeśli wartość jest inna, wykonaj instrukcje4 i zakończ blok switch:

Switch( wyrażenie ) {

Case wartość1:

Instrukcja1;

Break;

Case wartość2:

Instrukcja2;

Break;

Case wartość3:

Instrukcja3;

Break;

Default:

Intrukcja 4;

}

Przykład:

<?php

$kolor = "red";

switch ($kolor){

case "red":

echo "Dominującym kolorem jest czerwony<br>";

break;

case "blue":

Echo "Dominującym kolorem jest niebieski<br>";

break;

case "green":

echo " Dominującym kolorem jest zielony<br>";

default:

echo "Brak dominującego koloru<br>";

}

?>

**Operator warunkowy:**

Operator warunkowy działa podobnie jak instrukcja if. Zwraca wartość z jednego z dwóch wyrażeń rozdzielonych dwukropkiem:

Warunek ? wartość1 : wartość2

Jeżeli warunek jest prawdziwy, to jako wartość całego wyrażenia podstaw wartość1, w przeciwnym razie jako wartość całego wyrażenia podstaw wartość2.

**14. Pętle**

**For:**

Pętla typu for służy do budowania pętli, gdy został podany licznik jej wykonań oraz warunek, który musi być spełniony, aby kolejny raz wykonać pętle:

For ( wyrażenie początkowe; wyrażenie warunkowe; wyrażenie modyfikujące) {

Instrukcja;

}

* wyrażenie początkowe- inicjuje zmienną która jest używana jako licznik pętli
* wyrażenie warunkowe – określa warunek, który musi zostać spełniony, aby pętla została wykonana kolejny raz
* wyrażenie modyfikujące – modyfikuje zmienna która jest licznikiem

Przykład:

<?php

for ($i =1; $i <= 5; $i++) {

echo "Pętla wykonana $i raz/y <br>";

}

?>

**While:**

Blok instrukcji jest wykonywany w petli dopóki wyrażenie warunkowe jest prawdziwe.

While (warunek) {

Instrukcja;

}

Przykład:

<?php

$i = 0;

while ($i++ <5) {

echo "Petla wykonana $i razy <br>";

}

?>

**DO … While:**

Konstrukcja mowi: Wykonuj instrukcje dopóki wyrażenie warunkowe jest prawdziwe.

W tej petli blok instrukcji jest wykonywany co najmniej raz nawet jeśli warunek jest fałszywy.

Najpierw wykonywany jest ciąg instrukcji dopiero potem jest sprawdzany warunek.

Do {

Instrukcja;

}

While (warunek);

Przykład:

<?php

$i = 1;

do {

echo "Pętla wykana $i razy <br>";

}

while ($i++ < 5);

?>

**Pętla foreach:**

Pętla foreach umożliwia dostęp do elementow tablicy lub właściwości obiektu.

Foreach ($tablica as $wartość) {

Intrukcja;

}

LUB

Foreach ($tablica as $ klucz $wartość) {

Instrukcja;

}

Przykład:

<?php

$tab = array(

1 => 'biały',

2 => 'czarny',

3 => 'niebieski',

4 => 'zielony'

);

foreach ($tab as $x) {

echo "$x <br>";

};

?>

**Break:**

Instrukcja break służy do przerywania wykonywania pętli.

Przykład:

<?php

$i = 0;

while (true) {

echo "Wypisz $i <br>";

if ($i >= 20) break;

$i++;

}

?>

**Continue:**

Po jej napotkaniu nastepuje przerwanie wykonywania bieżącej iteracji i przejście na początek petli.

<?php

For ($i = 0; $i = <= 30; i$++ {

If (($i %3) != 0 ) continue;

Echo „$i; „;

}

?>