

PHP - podstawy

1. Podstawy składni języka PHP <ul style="list-style-type: none"> • każda instrukcja (nie blok instrukcji) kończy się średnikiem • wielkość liter ma znaczenie tylko w nazwach zmiennych • nazwy mogą składać się z liter, cyfr i znaku podkreślenia (_) • nazwy nie mogą zaczynać się od cyfry i nie mogą zawierać spacji 	
2. Dodawanie skryptu PHP w kodzie HTML <pre><?php // tu wstaw kod PHP ?></pre>	3. Komentarz jednolinijkowy <pre>// ta linia zostanie zignorowana # ta linia również zostanie zignorowana</pre>
4. Komentarz wielolinijkowy <pre>/* te linie zostaną zignorowane */</pre>	5. Blok instrukcji <pre>{ instrukcja1; instrukcja2; }</pre>
6. Umieszczanie tekstu na stronie <pre>echo "<h1>Tytuł</h1>"; // umieszcza tekst na stronie echo "<h1>", "Tytuł", "</h1>"; // poszczególne napisy można oddzielić przecinkiem print "napis"; // alternatywny sposób umieszczania tekstu</pre>	
7. Wartości <pre>5 // liczba całkowita (typ integer) -5 // ujemna liczba całkowita (typ integer) 3.14 // liczba rzeczywista (typ float) "napis" // tekst (typ string) '\$napis\$' // tekst, inny zapis, można umieszczać znaki specjalne (typ string) true // wartość logiczna, prawda (typ boolean) false // wartość logiczna, fałsz (typ boolean) null // wartość "pusta" (typ specjalny)</pre>	
8. Zmienne (definicja za pomocą znaku dolara \$ i znaku równości =) <pre>\$kod = "PHP"; // definicja zmiennej o nazwie \$kod i przypisanie do niej napisu "PHP" \$liczba = 10; // zmienna \$liczba przechowuje liczbę całkowitą 10 \$pi = 3.14; // zmienna \$pi przechowuje liczbę rzeczywistą 3,14 \$log = true; // zmienna \$log przechowuje wartość logiczną true (prawda)</pre>	
9. Stałe <pre>define("PI", 3.1415); // definicja stałej o nazwie "PI" i wartości 3,1415 define("EMAIL", "poczta@a.pl"); // definicja stałej "EMAIL" o wartości "poczta@a.pl"</pre>	
10. Operatory arytmetyczne <pre>\$a = 1 + 2; // dodawanie, zmienna \$a przechowuje wartość 3 \$b = 5 - 3; // odejmowanie, zmienna \$b przechowuje wartość 2 \$c = 2 * 5; // mnożenie, zmienna \$c przechowuje wartość 10 \$d = 4 / 2; // dzielenie, zmienna \$d przechowuje wartość 2 \$e = 5 % 2; // reszta z dzielenia, zmienna \$e przechowuje wartość 1 \$f = (2 + 3) * 4; // grupowanie (nawias), najpierw dodaj, potem pomnóż, wynik 20 \$g = 2 ** 4; // potęgowanie: 2⁴, zmienna \$g przechowuje wartość 16</pre>	
11. Operator konkatencji (łączenia napisów) – kropka <pre>\$a = "Tekst" . "
"; // zmienna \$a przechowuje napis "Tekst
"</pre>	

<p>12. Inkrementacja (zwiększanie o 1)</p> <pre>\$a++; // zwraca \$a, a następnie // zwiększa \$a o 1 // przykład: \$b = 5; echo \$b++; // wypisze: 5 echo \$b; // wypisze: 6 ++\$c; // zwiększa \$c o 1, // a następnie zwraca \$c // przykład: \$d = 5; echo ++\$d; // wypisze: 6 echo \$d; // wypisze: 6</pre>	<p>13. Dekrementacja (zmniejszanie o 1)</p> <pre>\$a--; // zwraca \$a, a następnie // zmniejsza \$a o 1 // przykład: \$b = 5; echo \$b--; // wypisze: 5 echo \$b; // wypisze: 4 --\$c; // zmniejsza \$c o 1, // a następnie zwraca \$c // przykład: \$d = 5; echo --\$d; // wypisze: 4 echo \$d; // wypisze: 4</pre>
<p>14. Operatory przypisania</p> <pre>\$a = 10; // do zmiennej \$a zostanie przypisana wartość 10 \$a += 2; // do zmiennej \$a zostanie dodane 2, wynik 12 \$a -= 4; // od zmiennej \$a zostanie odjęte 4, wynik 8 \$a *= 2; // zmienna \$a zostanie pomnożona przez 2, wynik 16 \$a /= 4; // zmienna \$a zostanie podzielona przez 4, wynik 4 \$a %= 3; // zmienna \$a przyjmie wartość reszty z dzielenia \$a przez 3, wynik 1 \$a .= "koniec"; // do zmiennej \$a na końcu zostanie dodany napis "koniec"</pre>	
<p>15. Przypisywanie długich napisów do zmiennej - heredoc</p> <pre>\$napis = "NAPIS"; \$tekst = <<<"TX" Bardzo długi \$napis TX; echo \$tekst; //wypisze: Bardzo długi NAPIS</pre>	<p>16. Przypisywanie długich napisów do zmiennej - nowdoc</p> <pre>\$napis = "NAPIS"; \$tekst = <<<'TX' Bardzo długi \$napis TX; echo \$tekst; //wypisze: Bardzo długi \$napis</pre>
<p>17. Operatory porównania</p> <pre>\$a == \$b; // równy, prawda jeśli \$a jest równe \$b \$a != \$b; // różny, prawda jeśli \$a jest różne od \$b \$a > \$b; // większy, prawda jeśli \$a jest większe od \$b \$a < \$b; // mniejszy, prawda jeśli \$a jest mniejsze od \$b \$a >= \$b; // większy lub równy, prawda jeśli \$a jest większe lub równe od \$b \$a <= \$b; // mniejszy lub równy, prawda jeśli \$a jest mniejsze lub równe od \$b \$a === \$b; // identyczny, prawda jeśli \$a równe \$b i są tego samego typu \$a !== \$b; // nieidentyczny, prawda jeśli \$a nie jest równe \$b lub nie są tego // samego typu</pre>	
<p>18. Operatory logiczne</p> <pre>\$a && \$b; // AND (i - koniunkcja), zwróci prawdę gdy \$a i \$b są prawdą \$a \$b; // OR (lub - alternatywa), zwróci prawdę gdy \$a lub \$b jest prawdą ! \$a; // NOT (nie - negacja), zwróci prawdę, jeśli \$a nie jest prawdą // NOT (nie - negacja), zwróci fałsz, jeśli \$a jest prawdą</pre>	
<p>19. Operator tłumienia błędów (@)</p> <pre>\$x = (5 / 0); // wyświetli BŁĄD, dzielenie przez 0 !!! \$x = @(5 / 0); // nadal błąd, ale bez wyświetlanie komunikatu</pre>	

<p>20. Tablice indeksowane numerycznie (typ array)</p> <pre>// definicja tablicy: \$auta = ["Audi", "Opel", "Kia"]; // dostęp do elementów tablicy: echo \$auta[0]; // wypisze: Audi echo \$auta[1]; // wypisze: Opel echo \$auta[2]; // wypisze: Kia // dodanie elementu na końcu tablicy: \$auta[] = "Ford" // sposób alternatywny definicji tablicy: \$auta = array("Audi", "Opel", "Kia"); // zliczanie elementów tablicy: echo count(\$auta); // wypisze: 3</pre>	<p>21. Tablice asocjacyjne (typ array)</p> <pre>// definicja tablicy: \$osoba = ["imie" => "Jan", "nazwisko" => "Kowalski", "email" => "poczta@a.pl"]; // dostęp do elementów tablicy: echo \$osoba["imie"]; //wypisze: Jan echo \$osoba["nazwisko"]; //wypisze: Kowalski echo \$osoba["email"]; //wypisze: poczta@a.pl // dodanie elementu do tablicy: \$osoba["wiek"] = 18; // sposób alternatywny definicji tablicy: \$osoba = array("imie" => "Jan, "nazwisko" => "Kowalski", "email" => "poczta@a.pl");</pre>
<p>22. Operator warunkowy</p> <pre>warunek ? wartość1 : wartość2; // jeżeli warunek jest prawdziwy // operator zwraca wartość1 // jeżeli warunek jest fałszywy // operator zwraca wartość2 // przykład: \$x = 10; \$wynik = (\$x < 0) ? "ujemna" : "dodatnia"; echo "Wartość zmiennej x jest \$wynik"; // wypisze: Wartość zmiennej x jest dodatnia</pre>	<p>23. Zmiana typu zmiennej</p> <pre>// tymczasowa zmiana typu zmiennej: (int) \$x; // rzutowanie do typu integer (float) \$x; // rzutowanie do typu float (string) \$x; // rzutowanie do typu string (bool) \$x; // rzutowanie do typu boolean // trwała zmiana typu zmiennej: settype(\$x, "int"); // zmiana na integer settype(\$x, "float"); // zmiana na float settype(\$x, "string"); // zmiana na string settype(\$x, "bool"); // zmiana na boolean</pre>
<p>24. Instrukcja warunkowa if</p> <pre>if (wyrażenie) { // kod do wykonania, jeżeli // wyrażenie jest prawdziwe } // przykład: \$x = 5; if (\$x == 5) { // wykonaj, echo "Pięć"; // jeżeli \$x równe 5 } // wypisze: Pięć</pre>	<p>25. Instrukcja warunkowa if ... else</p> <pre>if (wyrażenie) { // kod do wykonania, jeżeli // wyrażenie jest prawdziwe } else { // kod do wykonania, jeżeli // wyrażenie nie jest prawdziwe } // przykład: \$x = 5; if (\$x == 5) { // wykonaj, echo "Pięć"; // jeżeli \$x równe 5 } else { // wykonaj, jeżeli echo "Inna liczba"; // \$x różne od 5 } // wypisze: Inna liczba</pre>

26. Instrukcja warunkowa if ... elseif ... else

```

if (wyrażenie1) {
    // instrukcje do wykonania, jeżeli wyrażenie1 jest prawdziwe
}
elseif (wyrażenie2) {
    // instrukcje do wykonania, jeżeli wyrażenie2 jest prawdziwe,
    // i jednocześnie wyrażenie1 jest fałszywe
}
else {
    // instrukcje do wykonania, jeżeli wyrażenie1 i wyrażenie2 nie są prawdziwe
}

// przykład:
$x = 10;
if ($x == 5) {           // wykonaj, jeżeli $x równe 5
    echo "Pięć";
}
elseif ($x == 10) {    // wykonaj, jeżeli $x równe 10 i jednocześnie różne od 5
    echo "Dziesięć"
}
else {                 // wykonaj, jeżeli $x różne od 5 i od 10
    echo "Inna liczba";
}
// wypisze: Dziesięć

```

27. Instrukcja wyboru switch

```

switch ($zmienna) {

    case wartość1:
        // instrukcje do wykonania, jeśli
        // $zmienna równa wartość1
        break;

    case wartość2:
        // instrukcje do wykonania, jeśli
        // $zmienna równa wartość2
        break;
}

// przykład:
$a = 1;
switch ($a) {
    case 0:
        echo "Zero";
        break;
    case 1:
        echo "Jeden";
        break;
}
// wypisze: Jeden

```

28. Instrukcja wyboru switch ... default

```

switch ($zmienna) {

    case wartość1:
        // instrukcje do wykonania, jeśli
        // $zmienna równa wartość1
        break;

    case wartość2:
        // instrukcje do wykonania, jeśli
        // $zmienna równa wartość2
        break;

    default:
        // wykonaj, jeśli $zmienna będzie
        // różna od wartości1 i wartości2
}

// przykład:
$a = "arbuz"
switch ($a) {
    case "pomidor":
        echo "Pomidor";
        break;
    case "cebula":
        echo "Cebula";
        break;
    default:
        echo "Inne warzywo";
}
// wypisze: Inne warzywo

```

29. Pętla for

```

for (wyrażenie1; warunek; wyrażenie2) {
    // instrukcje pętli
}
// wyrażenie1 - instrukcje do wykonania przed rozpoczęciem pętli
// warunek     - jeżeli prawdziwy kolejne przejście pętli zostanie wykonane
// wyrażenie2  - instrukcje do wykonania po każdym przejściu (iteracji) pętli

// przykład:
for ($i = 0; $i < 10; $i++) {
    echo $i." ";
}
// wypisze: 0 1 2 3 4 5 6 7 8 9

```

30. Pętla while

```

while (wyrażenie) {
    // instrukcje do wykonania
    // w każdym przebiegu pętli,
    // jeżeli wyrażenie jest prawdziwe
}

// przykład:
$i = 0;
while ($i < 10) {
    echo $i." ";
    $i++;
}
// wypisze: 0 1 2 3 4 5 6 7 8 9

```

31. Pętla do ... while

```

do { // początek pętli
    // instrukcje pętli
} while (wyrażenie); //jeżeli wyrażenie jest
//prawdziwe skocz na
//na początek pętli

// przykład:
$i = 0;
do {
    echo $i." ";
    $i++;
} while ($i < 10);
// wypisze: 0 1 2 3 4 5 6 7 8 9

```

32. Instrukcja break

```

for ($i = 0; $i < 10; $i++) {
    if ($i == 5) { // jeżeli $i równe 5
        break;    // przerwij pętle
    }
    echo $i." ";
}
// wypisze: 0 1 2 3 4

```

33. Instrukcja continue

```

for ($i = 0; $i < 10; $i++) {
    if ($i == 5) { // jeżeli $i równe 5
        continue; // skocz do następnego
        // przebiegu pętli
    }
    echo $i." ";
}
// wypisze: 0 1 2 3 4 6 7 8 9

```

34. Pętla foreach – przeglądanie tablic numerowanych

```

foreach ($tablica as $element) {
    // instrukcje do wykonania
    // na każdym elemencie tablicy
    // Sposób działania:
    // za każdym przejściu pętli, pobierany
    // jest kolejny element tablicy $tablica
    // i przypisywany do zmiennej $element
}

// przykład:
$lata = [1999, 2001, 2002];
foreach ($lata as $rok) {
    echo $rok." ";
}
// wypisze: 1999 2001 2002

```

35. Pętla foreach – przeglądanie tablic asocjacyjnych

```

foreach ($tablica as $klucz => $wartosc) {
    // instrukcje do wykonania
    // na każdym elemencie tablicy
}

// przykład:
$osoba = ["imię" => "Jan",
         "nazwisko" => "Nowak",
         "wiek" => 20];
foreach ($osoba as $nazwa => $wartosc) {
    echo $nazwa."": ".$wartosc.", ";
}
// wypisze:
// imię: Jan, nazwisko: Nowak, wiek: 20,

```

<p>36. Funkcje</p> <pre>// definicja prostej funkcji: function hello() { echo "Cześć"; } // wywołanie funkcji: hello(); // wypisze: Cześć</pre>	<p>37. Przekazywanie argumentów do funkcji</p> <pre>// definicja funkcji dodającej // dwie liczby: function dodaj(\$a, \$b) { echo \$a + \$b; // wypisz sumę \$a i \$b } // wywołanie funkcji: dodaj(4, 2); // wypisze: 6 dodaj(2, 1); // wypisze: 3</pre>
<p>38. Zwracanie wartości przez funkcję - return</p> <pre>// definicja funkcji dzielącej // jedną liczbę przez drugą function podziel(\$a, \$b) { return \$a / \$b; // zwróć iloraz \$a i \$b } // wywołanie funkcji: echo podziel(8, 2); // wypisze: 4</pre>	<p>39. Argumenty funkcji przekazywane przez referencję</p> <pre>// definicja funkcji z argumentem // przekazywanym przez referencję function inkrementujA(&\$arg) { \$arg++; } // definicja funkcji z argumentem // przekazywanym przez wartość function inkrementujB(\$arg) { \$arg++; } \$liczba = 1; inkrementujB(\$liczba); echo \$liczba; // wypisze: 1 inkrementujA(\$liczba); echo \$liczba; // wypisze: 2</pre>
<p>40. Funkcje - zmienne globalne</p> <pre>\$liczba1 = 10; \$liczba2 = 20; function wyswietl() { global \$liczba1; echo \$liczba1; // wypisze: 10 echo \$liczba2; // błąd, \$liczba2 nie // jest widoczna w funkcji }</pre>	<p>41. Funkcje - zmienne statyczne</p> <pre>function inkrementuj() { static \$liczba1 = 0; \$liczba2 = 0; \$liczba1++; \$liczba2++; echo \$liczba1." ".\$liczba2; } inkrementuj(); // wypisze: 1 1 inkrementuj(); // wypisze: 2 1 inkrementuj(); // wypisze: 3 1</pre>
<p>42. Funkcje - argumenty domyślne</p> <pre>function dodaj(\$x, \$y = 1) { return \$x + \$y; } echo dodaj(2, 4); // wypisze: 6 echo dodaj(10); // wypisze: 11</pre>	