

[h]↳ Proměnné[/h]

Proměnné jsou nezbytnou součástí každého programovacího jazyka. Setkáte se s nimi téměř ve všech částech vašeho kódu. Představují v podstatě malé „úložiště“ jednotlivých dat, ať už číselných hodnot nebo třeba textových řetězců. Každá proměnná má svůj [b]název[/b], [b]hodnotu[/b] a [b]datový typ[/b].

[h]↳ Deklarace proměnných[/h]

[img]obrazek[/img]

Jak můžete vidět výše, proměnné v PHP začínají speciálním znakem \$ (dolar), za nimž těsně (tj. bez žádných mezer) následuje název proměnné (někdy se označuje jako [i]identifikátor[/i]), který podléhá hned několika pravidlům:

[ul]

[-]Název proměnné by měl obsahovat znaky a-z/A-Z a podtržítko, přičemž nesmí začínat čísly. Povoleno jsou pak i [url=http://php.net/manual/en/language.variables.basics.php]další znaky[/url], které se však v praxi téměř nepoužívají, neboť přináší velké množství nečekaných problémů.[-]

[-]Název proměnné musí být jedinečný. Pokud zadáte název, který již existuje, tak pouze pracujete se svou předchozí proměnnou.[-]

[-]V případě, že se nějaká proměnná nechová dle vašeho očekávání, je dobré si zkontrolovat její název, jestli není omylem pojmenována jako jiná proměnná.[-]

[-]Není vhodné ani tzv. „recyklovat“ proměnné. To znamená, že si definujete nějakou obecnou proměnnou např: [code]\$a[/code], a poté ji používáte k různým účelům v různých částech skriptu.[-]

[/ul]

Kromě těchto závazných pravidel, které při vytváření nové proměnné nesmíte porušit, existuje v PHP také řada konvencí. Co je to vlastně ta [i]konvence[/i]? Jedná se o různá doporučení či dohody ze strany programátorů. Zda se jich budete držet, je tedy pouze a čistě na vás.

[ul]

[-]Víceslovné názvy proměnných se dají zapsat hned několika způsoby, můžete například jednotlivé slova oddělovat podtržítkem (vice_slovna_promenna) nebo je spojit dohromady (viceslovnapromenna). V seriálu budeme preferovat způsob zápisu [i]lower-camelCase[/i] (viceSlovnaPromenna), protože je přehlednější než předchozí příklady.[-]

[-]Všechny proměnné by měly mít vhodné pojmenování, tak aby jejich název vystihoval podstatu

proměnné. Je lepší zvolit název „cz“ nebo „cenaZbozi“?[/-]

[-]Není vhodné používat ani příliš dlouhé názvy, protože kód je kvůli nim méně přehledný a při práci s nimi se často přepíšete.[/-]

[-]V neposlední řadě vás také jistě napadlo, který jazyk je pro pojmenovávání lepší – angličtina nebo čeština? I u této konvence záleží čistě na vašich osobních preferencích anebo na dohodě jednotlivých programátorů.[/-]

[/ul]

Poslední důležitou součástí každé nové proměnné je její hodnota (nebo-li uložená informace, někdy se nazývá také [i]literál[i]). Hodnotu k proměnné přiřadíte pomocí operátoru = „rovná se“. Prvotní přiřazení hodnoty k proměnné se odborně nazývá [i]inicializace[i].

[pre]

```
<?php
```

```
// Proměnná obsahuje celé číslo 23
```

```
$vek = 23;
```

```
// Proměnná obsahuje desetinné číslo 180.75 (záměrně s tečkou, viz. vysvětleno níže)
```

```
$vyska = 180.75;
```

```
// Proměnná obsahuje textový řetězec Jan Novák
```

```
$jmeno = "Jan Novák";
```

```
?>
```

[/pre]

[h]↳ Vypsání hodnot uložených v proměnných[/h]

Nyní máte vaše hodnoty uložené v označených proměnných a můžete s nimi libovolně manipulovat. Z kapitoly o [základech syntaxe](#) určitě víte, jak vypisovat jednotlivé hodnoty do výsledného kódu stránky. U proměnných to není o mnoho těžší, nejlépe si to zase vyzkoušíte v následujícím příkladě:

[pre]

```
<?php
```

```
// Pakliže jste výše správně inicializovali tyto proměnné, můžete je vypsát
```

```
echo $vek; // vypíše „23“
```

```
echo $vyska; // vypíše „180.75“
```

```
// Přijdete na to, proč se tento příkaz neprovede podle očekávání?
```

```
echo $Vyska;
```

```
?>
```

[/pre]

V uvedených příkladech vždy při výpisu proměnné dostanete předpokládaný výsledek až na poslední příklad. V čem se liší? Místo desetinného čísla 180.75 se vám zobrazí poznámka (Notice): [i]„Undefined variable: Vyska in ...“[/i]. Možná již někteří z vás správně zaregistrovali různorodost velikosti písmen u proměnné, z toho tedy vyplývá jedno obecné pravidlo. Pokud manipulujete s proměnnými, musíte striktně dodržovat jejich přesný název včetně velikosti písmen.

Kromě vypisování jednotlivých hodnot, můžete začlenit proměnné také rovnou do řetězců.

[pre]

```
<?php
```

```
// Proměnné obsažené v jednoduchém textovém řetězci
```

```
echo "Zákazník pan $jmeno měří $vyska cm.";
```

```
?>
```

[/pre]

Výstupem v takovém případě bude: [i]„Zákazník pan Jan Novák měří 180.75 cm“[/i]. Všimněte si, že řetězec je uzavřený do uvozovek a to z toho důvodu, že dolar se zde chová jako speciální symbol, a proto se proměnné zpracují. V případě, že nechcete z jakéhokoliv důvodu vypsát proměnné, musíte dolar escapovat nebo uzavřít řetězec do apostrofů.

[pre]

```
<?php
```

```
// Řetězec uzavřený do apostrofů
```

```
echo 'Zákazník pan $jmeno měří $vyska cm.'; // vypíše „Zákazník pan $jmeno měří $vyska“
```

```
// Speciální symbol dolaru je v řetězci escapován
```

```
echo "Zákazník pan \$jmeno měří \$vyska cm."; // vypíše „Zákazník pan $jmeno měří $vyska“
```

```
?>
```

[/pre]

[h]↳ Datové typy[/h]

Datové typy reprezentují druh zadané hodnoty v proměnné. Jelikož je PHP

[url=http://cs.wikipedia.org/wiki/Typov%C3%A1_kontrola]dynamicky typovaný jazyk[/url], závisí datový typ na uvedené hodnotě. V praxi to znamená, že pokud máte uloženou např. hodnotu 5, datový typ proměnné bude číselný. PHP umí pracovat celkem se čtyřmi základními ([i]skalárními[/i]) datovými typy.

Nutno podotknout, že datový typ nemůžete při běžném výpisu vidět (zobrazí se pouze hodnota proměnné), proto pokud potřebujete zjistit datový typ (tzv. „vypsát proměnnou“) nepoužívejte

typickou jazykovou konstrukci echo, ale speciální funkci na to určenou – `var_dump`.

```
[pre]
<?php
$promenna = 0x2A;

// užití funkce var_dump
var_dump($promenna); // vypíše „int(42)“
?>
[/pre]
```

Funkce `var_dump` vám vždy u správně inicializovaných proměnných vypíše datový typ a hodnotu (u textových řetězců také počet znaků, který ale není vždy přesný viz. PHP manuál). U výše zmíněného příkladu tak víme, že hodnota proměnné je 42 a datový typ integer.

boolean (logický typ) – datový typ boolean nabývá pouze hodnoty `true` a `false`. Často se s ním můžete setkat například při určování pravdy (true) a nepravdy (false).

```
[pre]
<?php
// Proměnná s datovým typem boolean a hodnotou true (ano, PHP je super)
$phpJeSuper = true;

// Proměnná s datovým typem boolean a hodnotou false (ne, PHP není super)
$phpJeSuper = false;
?>
[/pre]
```

integer (celé číslo) – zkráceně `int` uchovává kladná i záporná čísla včetně nuly o 32 bit. nebo 64 bit. rozsahu (závisí na platformě na které server běží). Nejvyšší možné číslo lze zjistit pomocí `echo PHP_INT_MAX;`, nejnižší možné je pak o 1 nižší s negativním znaménkem. Pro představu na 32 bit. platformě je to číslo 2147483647. Při přesažení povoleného rozsahu dojde k tzv. přetečení (overflow), což znamená, že číslo se přetypuje do datového typu float (ten má ještě větší rozsah) a příliš velký float se změní na INF (infinity = nekonečno).

```
[pre]
<?php
// číslo 9 223 372 036 854 775 808, dochází k přetečení
$cislo = 9223372036854775808;
```

```
// jak můžete vidět, jedná se při výpisu proměnné již o datový typ float
var_dump($cislo); // vypíše hodnotu proměnné 9.2233720368548E+18 a datový typ float
?>
[/pre]
```

Čísla lze kromě v desítkové soustavě zapsat také v osmičkové, šestnáctkové nebo dvojkové soustavě (od PHP 5.4). V případě, že máte nižší verzi PHP se vám místo zápisu čísla v dvojkové soustavě zobrazí parse error.

```
[pre]
<?php
// číslo (47) zapsané v obvyklé desítkové soustavě s datovým typem int
$desitkovaSoustavaPlus = 47;

// číslo (-47) zapsané v obvyklé desítkové soustavě s datovým typem int
$desitkovaSoustavaMinus = -47;

// číslo (47) zapsané v šestnáctkové soustavě s datovým typem int
$sestnackovaSoustava = 0x2F;

// číslo (47) zapsané v osmičkové soustavě s datovým typem int
$osmickovaSoustava = 057;

// číslo (47) zapsané v dvojkové soustavě s datovým typem int
$dvojkovaSoustava = 0b101111;

echo $desitkovaSoustavaPlus; // vypíše: „47“
echo $desitkovaSoustavaMinus; // vypíše: „-47“
echo $osmickovaSoustava; // vypíše „47“
echo $sestnackovaSoustava; // vypíše „47“
echo $dvojkovaSoustava; // vypíše „47“
?>
[/pre]
```

U číselných soustav (šestnáctková, osmičková, dvojková) si můžete všimnout tzv. prefixu (označen červeně), právě ten určuje o jakou soustavu se jedná. Pokud chcete například zapsat číslo v šestnáctkové soustavě, stačí před samotné číslo uvést prefix „0x“. Dávejte si zejména pozor na osmičkovou soustavu – vždy když je před číslem uvedena nula (např: 0123, 057) jedná se o zapsání čísla v osmičkové soustavě.

[b]float (desetinné číslo)[/b] – datový typ `float`, někdy se označuje také jako `double` nebo reálné číslo, se zabývá čísly s plovoucí desetinnou čárkou. I když se v češtině užívá desetinná čárka, v jazyku PHP (dle angličtiny) musíte uvádět desetinnou tečku.

```
[pre]
<?php
// správně zapsané desetinné číslo (4,7) s datovým typem float
$realneCislo = 4.7;

var_dump($realneCislo); // vypíše hodnotu 4,7 a datový typ float

// špatně zapsané desetinné číslo (4,7), způsobí parse error
$realneCislo = 4,7;
?>
[/pre]
```

Desetinná čísla datového typu float mají omezenou přesnost, proto je používejte s rozvahou. Podívejte se kupříkladu na následující příklad:

```
[pre]
<?php
// menší matematický příklad, u kterého předpokládáme výsledek 8
$priklad = (0.1+0.7)*10;
?>
[/pre]
```

Věřte nebo ne, ale vnitřním výsledkem v PHP nebude osmička, ale číslo „7.9999999999999991118 ...“, což může způsobit například problémy při porovnávání se skutečnou osmičkou. Proč k této nepřesnosti dochází? Představte si libovolný interval s reálnými čísly. Těchto čísel bude nekonečně mnoho, nicméně počítače mají omezenou paměť, a tak nemohou uložit přesně všechna čísla. Proto datový typ float uloží vždy nejbližší číslo, které uložit dokáže a vzniká tak nepřesnost.

[b]string (řetězec)[/b] – datový typ `string` slouží pro ukládání řetězců, které mohou obsahovat libovolné informace (čísla, písmena, znaky z ASCII tabulky apod.). U tohoto typu se přiřazovaná hodnota k proměnné uzavírá do uvozovek nebo apostrofů.

```
[pre]
<?php
// Proměnná obsahuje textový řetězec
$text = "Krátký textový řetězec";

var_dump($text); // vypíše řetězec „Krátký textový řetězec.“ a datový typ string
?>
[/pre]
```

Do řetězce nevkládejte hodnoty ostatních datových typů, pokud to není nezbytně nutné. V obecné rovině platí zejména u čísel následující pravidlo - pokud budete s čísly provádět matematické operace, neuzavírejte je do apostrofů nebo uvozovek. Jiná situace nastává například u PSČ nebo telefonních čísel.

```
[pre]
<?php
// u tohoto příkladu provádíme aritmetickou operaci sčítání
$priklad = 5+5;

var_dump($priklad); // vypíše datový typ integer a hodnotu proměnné 10

// PSČ obsahuje mezeru mezi čísly, takže jej nemůžeme zapsat jako číslo
$psc = "110 00";

// telefonní číslo by se sice dalo zapsat bez uvozovek, ale není to "typické" číslo
$telCislo = "775357891";
?>
[/pre]
```

[h]↳ Další vlastnosti proměnných[/h]

Možná vás u předchozích příkladů napadla jedná myšlenka. Co se stane, když proměnné nepřidáme žádnou hodnotu? Tuto možnost demonstduje následující příklad:

```
[pre]
<?php
// neinicializovaná proměnná
$promenna;

// vypsaní proměnné
var_dump($promenna); // vypíše „NULL“
?>
[/pre]
```

V první řadě se zobrazí NOTICE o nedefinované proměnné a `var_dump` vám vypíše `NULL`. Co to znamená? `NULL` je speciální hodnota (a zároveň pravoplatný datový typ), která reprezentuje proměnné bez hodnoty (neinicializované proměnné).

```
[pre]
<?php
$promenna = NULL;
```

```
// proměnnou můžete také inicializovat později
$promenna = 5;
```

```
var_dump($promenna); // vypíše „int(5)“
?>
[/pre]
```

Kromě neinicializované proměnné rozlišujeme v PHP i proměnnou tzv. „prázdnou“.

```
[pre]
<?php
// Tato proměnná má prázdnou hodnotu datového typu string
$promenna = "";

var_dump($promenna); // vypíše „string(0) ""
?>
[/pre]
```

S prázdnými/neinicializovanými proměnnými souvisí také [i]jazykové konstrukty[/i] (klíčová slova) [b>isset[/b], [b>empty[/b] a [b>unset[/b]. Pomocí prvních dvou jazykových konstruktů zjistíte, jestli proměnná existuje, či zda je prázdná. Při výpisu získáte logické hodnoty [code>true[/code]/[code>false[/code].

```
[pre]
<?php
$promenna = NULL;
$cislo = 5;

var_dump(isset($promenna)); // vypíše „bool(false)“

var_dump(isset($cislo)); // vypíše „bool(true)“

var_dump(empty($promenna)); // vypíše „bool(true)“

var_dump(empty($cislo)); // vypíše „bool(false)“
?>
[/pre]
```

Z uvedeného příkladu si můžete odvodit, že [code>isset[/code] určuje, zda proměnná existuje a není [code>NULL[/code]. [b>Empty[/b] může být zase nápomocné při určení, zda je proměnná prázdná (seznam hodnot, které jsou považovány za prázdné naleznete v PHP manuálu). Poslední jazykový konstrukt [code>unset[/code] zruší uvedené proměnné.

```
[pre]
<?php
$textovyRetezec = "Textový řetězec";

// Zrušení proměnné textovyRetezec
unset($textovyRetezec);
?>
[/pre]
```

Proměnné číslo a `$textovyRetezec` v současné době neexistují. Pro zrušení proměnných se nedoporučuje přiřazovat hodnotu `NULL` k proměnnám, vždy je lepší použít jazykový konstrukt `unset`.

U již vytvořených proměnných je možné jejich hodnotu také přepsat a to dokonce napříč všemi datovými typy. Vytvořte si proměnnou s názvem `$text`, která obsahuje libovolný řetězec. Poté hodnotu přepište číslem 6.

```
[pre]
<?php
// Původní proměnná s datovým typem string
$text = "Textový řetězec";

// Přepsání původní hodnoty
$text = 6;

// nějaký další kód ...

// Znovu přepsání původní hodnoty
$text = 54;
?>
[/pre]
```

Tento jev se nazývá tzv. *přetypování*, které se bude podrobně probírat v další kapitole. Důležitý je zde zejména fakt, že všechny proměnné mohou (z)měnit svou hodnotu.

Nejčastější chyby

Stejně jako v ostatních jazycích i v PHP existuje řada častých chyb, které zejména začátečníci často dělají ve svém kódu. V níže uvedeném textu naleznete několik takových příkladů, které jsou buď syntakticky nebo i logicky špatně.

```
[pre]
<?php
```

```
// Tento příklad skončí parse errorem, první znak v názvu proměnné nesmí být číslo
$6cislo = 6;

// Tento příklad skončí parse errorem, proměnná obsahuje ve svém názvu nepovolené znaky
$ci:slo = 6;

// proměnná „retezec“ neexistuje, proto nedojde k výpisu
// zobrazí se chybové hlášení NOTICE: Undefined variable ...
echo $retezec;

// U desetinného čísla se píše tečka nikoliv čárka! Příklad skončí parse errorem.
$desetinneCislo = 6,5;

// tohle číslo není datového typu int, nýbrž je to pravoplatný string!
$cisloInteger = "5";

// Tato hodnota je zapsána v osmičkové soustavě - namísto „0526“ dostanete u výpisu číslo „342“.
$hodnota = 0526;
echo $hodnota;

// Tento příklad není vyložená chyba, ale název proměnné je zvolen značně nečitelně
$telesovauhloprickakrychledelkacm = 20;
?>
[/pre]
```

Další podobné příklady si jistě na základě svých znalostí z předchozího textu dokážete domyslet.

[h]↳ Proměnné proměnné[/h]

V jazyce PHP se někdy můžete setkat s exotickým přístupem k proměnné, který se oficiálně nazývá [i]proměnné proměnné[/i] (v angličtině: variable variables). Tento způsob v podstatě nemá praktické využití, slouží spíše jako kuriozita, kdyby jste se někdy s podobným přístupem setkali.

```
[pre]
<?php
// vytvoření nové proměnné
$novaPromenna = "nazevPromenne";

// inicializujeme novou proměnnou
$nazevPromenne = 5;
```

```
// netradiční způsob výpisu proměnné  
echo $$novaPromenna; // vypíše „5“  
?>  
[/pre]
```

Proměnné proměnné také v žádném případě nezaměňujte s poli, o kterých se dozvíte v některé z budoucích kapitol.

[h]↳ Vyzkoušejte si[/h]

- 1) Vytvořte si několik proměnných a přiřadte jim různé hodnoty. Vyzkoušejte si, které chyby se objeví, pokud nedodržíte některé z pravidel.
- 2) Všechny vytvořené proměnné si vypíšte do kódu stránky.