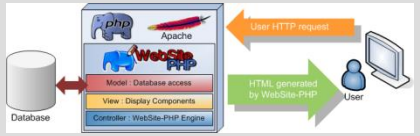
Internet a zdravotnická informatika

Přednáška 2: Základy PHP

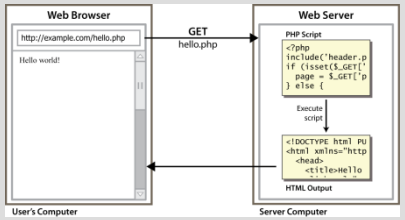
Základní pojmy a historie PHP

Skriptování na serveru (Server-side scripting)
Technologie, aplikovaná na webovém serveru, která umožňuje, aby požadavek uživatele na www stránku byl doplněn o obsah, vygenerovaný na serveru.

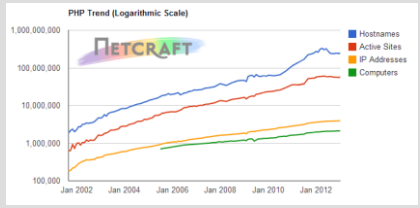
PHP (rekurzivní zkratka PHP: Hypertext Preprocessor, „PHP: Hypertextový preprocesor", původně Personal Home Page) je skriptovací programovací jazyk, určený především pro programování **dynamických internetových stránek**. Nejčastěji se začleňuje **přímo do struktury jazyka HTML, XHTML či WML**.



V kombinaci s operačním systémem Linux, databázovým systémem (obvykle MySQL nebo PostgreSQL) a webovým serverem Apache je často využíván k tvorbě webových aplikací. Pro tuto kombinaci se vžila zkratka **LAMP** – tedy spojení Linux, Apache, MySQL a PHP nebo Perl. Alternativou je **WAMP**, analogicky běžící na OS Windows.



Jazyk PHP byl původně vytvořen Rasmusem Lerdorfem v roce **1995** a jeho vývoj pokračuje do současnosti. PHP patří mezi tzv. **free software** publikovaný pod PHP licencí, která je kompatibilní s GNU General Public License (GPL).



V roce 2013 odhadla organizace Netcraft počet webů, které běží na PHP na více než **200 milionů**. Podle W³Techs běží PHP na **77,4 % webů** (říjen 2022). Mezi weby, které běží na PHP jsou **Facebook.com, Wikipedia.org, Whatsapp.com, Tumblr.com**, **Flickr.com**, ad.

Přednáška 2: Základy PHP

Jak PHP funguje?

1. **Prohlížeč odešle na server požadavek na PHP soubor.**
2. **PHP interpret provede PHP kód a vygeneruje HTML.**
3. **Server odešle zpět prohlížeči odpověď s tímto kódem.**
4. **Prohlížeč podle HTML kódu zobrazí WWW stránku.**

Přednáška 2: Základy PHP

Psaní PHP kódu

Soubor s kódem
PHP kód se píše přímo do HTML kódu webové stránky; to, že stránka obsahuje php kód oznamujeme webovému serveru příponou *.php.

Oddělení kódu
Ve skriptu samotném je kód php uzavřen dvojicí značek:

```
<?php echo 'ahoj světe'; ?>
```

resp.

```
<? echo 'ahoj světe'; ?>
```

příp.

```
<?= 'ahoj světe'; ?>
```

Oddělení instrukcí
Jednotlivé instrukce se v jazyce PHP oddělují středníkem:

```
<?php
echo "První řádek";
echo "Druhý řádek";
?>
```

Pozn.:
Před poslední instrukcí nemusí středník být. Je ale lepší jej tam psát.
Na rozdíl od některých jiných jazyků jsou konce řádků pro PHP nepodstatné. Nepodstatné jsou rovněž mezery a tabelátory.

Přednáška 2: Základy PHP

Komentáře; Příkaz echo

Komentáře

Komentáře umožňují kód opatřit poznámkami. Podporuje to srozumitelnost kódu a napomáhá jeho čtení.

```
<?php
echo 'Ahoj světe!'; // vypíše pozdrav
?>
```

Příp. víceřádkový komentář

```
<?php
/* vypíše pozdrav
pomocí příkazu echo */
echo 'Ahoj světe!';
?>
```

Escape sekvence (viz. např Escapování – definitivní příručka)

Sekvence	Význam
\n	Nový řádek
\"	Uvozovky
\r	Návrat vozíku
\t	Tabelátor
\\	Zpětné lomítko
\$	Dolar

Příkazy

Příkaz echo

Vytiskne jeden nebo více řetězců:

```
<?php
echo 'Ahoj světe', ' dnes je krásně!';
?>
```

Rozlišujte: a)

```
<?php
echo "Ahoj světe", "\ndnes je krásně!";
?>
```

b)

```
<?php
echo "Ahoj světe", "<br>dnes je krásně!";
?>
```

Přednáška 2: Základy PHP

Spojování řetězců a proměnných

Spojování řetězců

Spojování retězců se provádí pomocí operátoru „.“ (tečka):

```
<?php
echo "Ahoj světe, " . "dnes je krásně!";

echo 'Ahoj světe, ' .' dnes je krásně!';
?>
```

Concatenation		
Sequence	**When**	**Example**
""	Single variable	$myString = "Adding single variable $string to this string";
.	Single line	$myString = 'Combining multiple ' . $strings . ' on one line';
.=	Multiple lines	$myString = 'Combining multiple lines '; $myString .= $strings; $myString .= ' by using the assignment operator';

Spojování řetězců a proměnných

Opět je možné použít operátor „.“:

```
<?php
$den = ”patek”;
echo "Ahoj světe, " . "dnes je " . $den;
?>
```

Nebo je možné (pouze u dvojitých uvozovek) operátor vynechat:

```
<?php
echo "Ahoj světe, " . " dnes je {$den}";
?>
```

Složené závorky je vhodné používat u složitějších proměnných nebo tam, kde by z kontextu nebylo jasné, co je proměnná a co řetězec.

```
<?php
echo "Ahoj světe, " . " dnes je {$tyden[4]}";
?>
```

Přednáška 2: Základy PHP

Proměnné (1/2)

Proměnné

Typ proměnné v PHP se určuje v okamžiku **přiřazení hodnoty** do proměnné. Nicméně během programu může proměnná svůj typ změnit, ať už díky instrukci v kódu nebo v důsledku nějakého výpočtu. To se pak nazývá **přetypování**.

Datový typ	Název v PHP	
Celá čísla	Integer	Uchovává celá **kladná i záporná čísla** (a nulu) od cca -2 biliónů po + 2 bilióny (2^{32} čísel)
Desetinná čísla	Float, Real	Uchovává **desetinná čísla**. S přesností většinou na 14 desetinných míst a rozsahem, který začíná jedničkou a může mít 308 nul.
Řetězce	String	Uchovává **texty neboli řetězce**. Řetězec je znak nebo sada znaků, v PHP prakticky neomezené délky.
Logický typ	Boolean	Uchovává hodnotu "pravda" nebo "nepravda". Zapisuje se jako **true** a **false**.

Přednáška 2: Základy PHP

Proměnné (2/2); konstanty

Každá proměnná musí mít **jednoznačný název**. Ten v PHP začíná znakem „**$**" (znak dolaru) a následuje (bez mezery) **jménem proměnné**. První znak toho pojmenování musí být buď písmeno „a-z" nebo podtržítko. Nesmí to být číslo ani nic jiného.
Názvy proměnných v PHP **rozlišují mezi malými a velkými písmeny**. V praxi se proměnné na rozdíl od konstant píší malými písmeny. V názvech proměnných lze použít české znaky včetně diakritiky; běžně se to ale nedělá. Desetinná čísla se **zadávají s tečkou**, ne s čárkou. Řetězce se uzavírají do **uvozovek** nebo do **apostrofů**.
Proměnné přiřadíme nějakou hodnotu pomocí znaku „**=**" (rovná se).
Příklady:

```
<?php
$num_rows = 10; // celé číslo
$cena = 22.00; // desetinné číslo
$text = 'Ahoj světe!'; // řetězec
$odpoved = true; // logická hodnota
$4patra = 'čtyřpatrový'; // špatně: proměnná nesmí začínat číslem
$rok = 2009;
echo $Rok; // nevypíše nic, nesouhlasí velikost písmen
?>
```

Konstanty (nezačínají symbolem „$" a píší se velkými písmeny)
Příklady:

```
<?php
define('IZI', 'Internet a zdravotnická informatika');
?>
```

Přednáška 2: Základy PHP

Funkce pro práci s proměnnými

isset()
Funkce isset() se používá pro zjišťování, zda byla proměnné přiřazena hodnota.
Příklad:

```
<?php
$prom = 1;
if (isset($prom)) echo $prom; // zobrazí hodnotu proměnné
?>
```

empty()
Funkce empty() vrací true, pokud nebyla proměnné přiřazena hodnota, nebo má hodnotu rovnou 0 nebo hodnotu prázdného řetězce.
Příklad:

```
<?php
$text = 'Ahoj světe!';
$prom = 1;
if (empty($text)) echo 'prázdný řetězec'; // nevypíše nic, řetězec není prázdný
if (!empty($prom)) echo $prom; // zobrazí hodnotu proměnné
?>
```

is_int() a is_string()
Funkce vracejí true, pokud je proměnná celým číslem, resp. řetězcem.
Příklad:

```
<?php
$prom = 1; if (is_int($prom)) echo 'funkce je celé číslo';
?>
```

Přednáška 2: Základy PHP

Automatické a explicitní přetypování proměnných

Každá proměnná v PHP má svůj datový typ. Ten je určen **automaticky přiřazenou proměnnou**.

Příklad:

```
<?php
$prom1 = 1; // proměnná je celé číslo
$prom2 = 1.0; // proměnná je desetinné číslo
?>
```

Automatické přetypování nastává např. po provedení nějakého výpočtu.

Příklad:

```
<?php
// proměnná je celé číslo
$prom3 = 2;
// proměnná je desetinné číslo
$prom3 = $prom3/2;
?>
```

K explicitnímu přetypování se používá funkce **settype()**. Pro zjišťování typu proměnné pak funkce **gettype()**. Vypsat typ proměnné lze také pomocí funkce **var_dump()**.

Nový typ (vpravo)-> Původní typ (dole)	String	Integer	Float	Boolean
String	-	Začíná-li číslem, pak toto číslo, jinak nula.	Desetinné číslo musí zabírat celý řetězec, jinak celé číslo, resp. nula.	"" a "0" =false, ostatní true
Integer	Převede na řetězec	-	Bez problému	0=false, ostatní true
Float	Převede na řetězec	První integer směrem k nule	-	0.0=False, ostatní True
Boolean	false="" true="1"	false=0 true=1	false=0.0 true=1.0	-

Internet a zdravotnická informatika

Přednáška 2: Základy PHP

Operátory (1/3)

Operátory umožňují kombinovat hodnoty proměnných za účelem získání nové hodnoty.

Aritmetické operátory

Symbol	Název	Příklad	Výsledek
+	sčítání	<?php echo 7 + 5; ?>	12
-	odečítání	<?php echo 7 – 5; ?>	2
*	násobení	<?php echo 7 * 5; ?>	35
/	dělení	<?php echo 7 / 5; ?>	1.4
%	zbytek (modulo)	<?php echo 7 % 5; ?>	2

Operátor přiřazení

```
<?php
$a = $b;
$a = $a + 3;
$a += 3; // operátor zkráceného přiřazení ?>
```

Inkrementální / dekrementální operátor

```
<?php
$x = 4;
echo $x++; // výraz je vyhodnocen před tím, než byla hodnota inkrementována
$x = 4;
echo ++$x; ?>
```

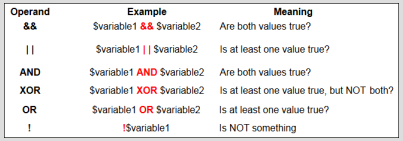
Internet a zdravotnická informatika

Přednáška 2: Základy PHP

Operátory (2/3)

Operátory porovnání
Používají se pro **testování podmínek**. Výrazy, používající tyto operátory, budou vždy vyhodnoceny jako **true** nebo **false**.

Symbol	Význam	Příklad	Bude vyhodnoceno jako true, když
==	rovnost	$i == $j	$i a $j mají stejnou hodnotu
<	menší	$i < $j	$i je menší než $j
>	větší	$i > $j	$i je větší než $j
<=	menší nebo rovno	$i <= $j	$i je menší nebo rovno než $j
>=	větší nebo rovno	$i >= $j	$i je větší nebo rovno než $j
!=	nerovnost	$i != $j	$i se nerovná $j
<>	nerovnost	$i <> $j	$i se nerovná $j



Operand	Example	Meaning
&&	$variable1 && $variable2	Are both values true?
||	$variable1 || $variable2	Is at least one value true?
AND	$variable1 AND $variable2	Are both values true?
XOR	$variable1 XOR $variable2	Is at least one value true, but NOT both?
OR	$variable1 OR $variable2	Is at least one value true?
!	!$variable1	Is NOT something

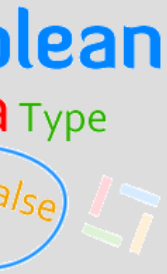
Logické operátory

Symbol	Význam	Příklad	Bude vyhodnoceno jako true, když
&&	a (and, logický součin)	$i && $j	budou $i a $j obě vyhodnoceny jako pravda
||	nebo (or, logický součet)	$i || $j	$i, $j nebo obě budou vyhodn. jako pravda
!	negace	!$i	bude $i vyhodnoceno jako nepravda
xor	exkluzivní or	$i xor $j	bude právě jedna hodnota pravdivá

Přednáška 2: Základy PHP

Operátory (3/3)

Příklady výrazů s logickými operátory

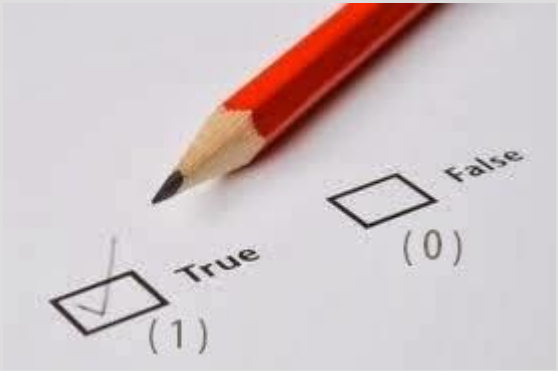


Logický součin

```
<?php
// jestliže například
$mam_papir=true; $mam_tuzku=true;
// ... je jasné, že
$mohu_psat=$mam_papir && $mam_tuzku;
?>
```

Logický součet

```
<?php
// jestliže například
$mam_auto=true; $mam_kolo=true;
// ... je jasné, že
$mohu_jezdit=$mam_auto || $mam_kolo;
?>
```



Negace

```
<?php
// jestliže například
$mam_hodne_penez = true;
// a pokud
$jsem_bohaty = !$mam_hodne_penez;
// ... je jasné, že
var_dump($jsem_bohaty); // false
?>
```

Exkluzivní logický součet (XOR)

```
<?php
// jestliže například
$jedu_autem=true; $jdu_pesky=false;
// ... je jasné, že
$dostanu_se_do_cile=($jedu_autem xor $jdu_pesky);
var_dump($dostanu_se_do_cile); // pokud jen jedno je true
?>
```

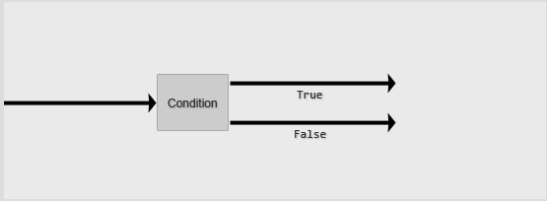
Přednáška 2: Základy PHP

Příkazy (1/3)

Příkazy pro větvení programů (podmínky)

Podmínky fungují v tom nejjednodušším případě tak, že nejprve je vyhodnocen určitý výraz. Je-li výraz pravdivý, provede se příkaz. PHP má k dispozici dvě podmíněné konstrukce.

Příkaz if



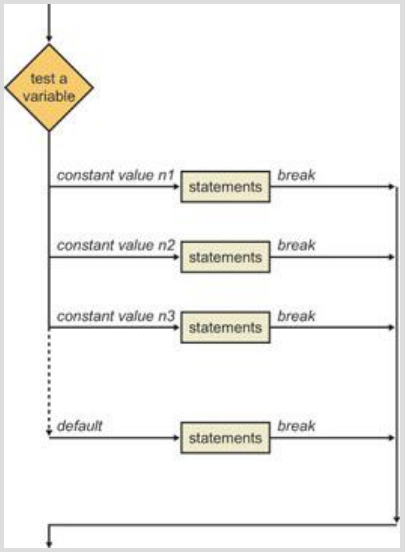
Příklad:

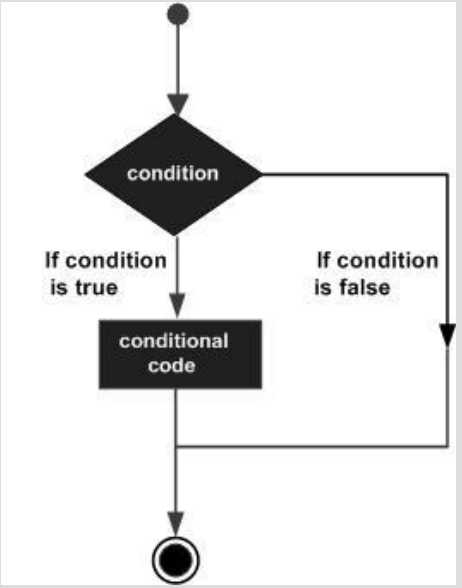
```
<?php
$prom2 = 1;
if (isset($prom)) echo $prom;
if (isset($prom)) echo $prom; else echo 'Proměnná nebyla nastavena.';
if (isset($prom)) echo $prom; elseif ($prom2 == 1) echo 'Proměnná prom2 má hodnotu 1.';
?>
```

Příkaz switch

Příklad:

```
<?php
$string = 'Ahoj světe!';
switch $string {
                case 'Ahoj studenti!':
                                echo $string;
                                break;
                case 'Ahoj světe!':
                                echo $string;
                                break;
                default:
                                echo 'Ani jedno z toho';
}
```





Internet a zdravotnická informatika

Přednáška 2: Základy PHP

Příkazy (2/3)

Cykly

Příkazy cyklů jsou v každém programovacím jazyce jedním z nejdůležitějších prvků. Cykly jsou určeny k opakovanému provádění bloku kódu v zadaném počtu nebo do splnění podmínky. PHP rozlišuje tři typy cyklů:

Cykly while

Cyklus while je nejjednodušším příkladem cyklu, jeho syntaxe je podobná podmínce if.

Příklad:

```
<?php
$day = 1;
while ($day < 5) {
        echo 'Den č. ' . $day . ' - nemáme IZI' . '<br />';
        $day++;
}
?>
```

Provádění cyklu lze přerušit pomocí **continue** nebo **break**.

Příklad:

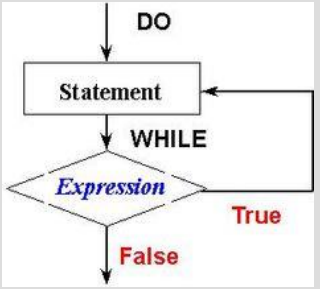
```
<?php
$day = 7;
while (--$day) {
        if ($day == 5) {
                echo 'Den č. ' . $day . ' - máme IZI' . '<br />';
                break;
        }
} ?>
```

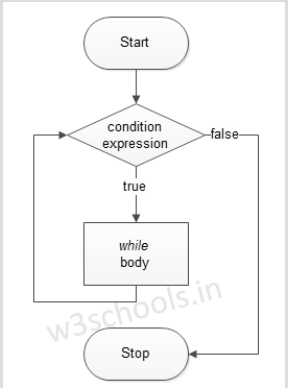
Cykly do...while

Podmínka je testována na konci každé iterace.

Příklad:

```
<?php
$day = 1;
do {
        echo 'Den č. ' . $day . ' - nemáme IZI' .
'<br />';
} while (++$day < 5)
?>
```





Přednáška 2: Základy PHP

Příkazy (3/3)

Cykly for

Syntaxe cyklů **for** je trochu složitější, nicméně jsou tyto cykly někdy vhodnější než cykly while.

Příklad:

```
<?php
for ($i=1; $i<8; $i++) {
            echo $i . '<br />';
}
?>
```

Cyklus foreach

Tento cyklus umožňuje iterovat nad **poli**, tzn. procházet předem definovaná pole. Cyklus funguje pouze nad proměnnou s datovým typem pole. Má dvě syntaxe.

Jednodušší syntaxe:

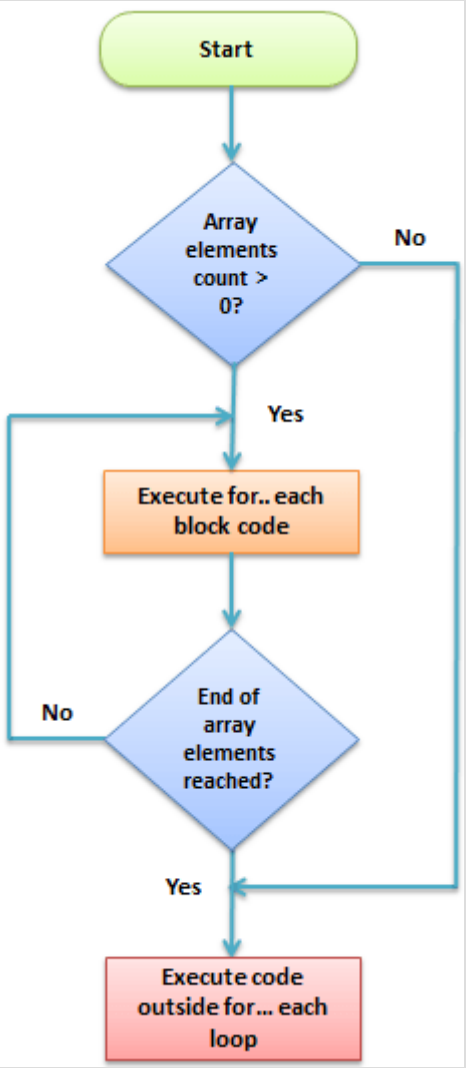
Příklad:

```
<?php
$arr = array(1, 2, 3, 4);
foreach ($arr as $value) {
    echo $value . '<br />';
}
?>
```

Složitější syntaxe s klíčem:

Příklad:

```
<?php
$arr = array(1, 2, 3, 4);
foreach ($arr as $key => $value) {
    echo $key . ' => ' . $value . '<br />';
}
?>
```



Internet a zdravotnická informatika

Přednáška 2: Základy PHP

Pole (1/2)

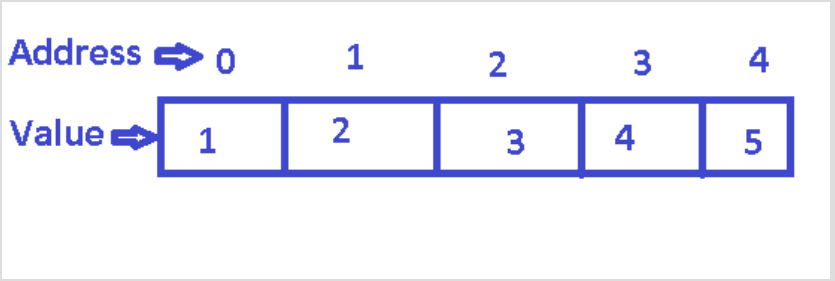
Pole

Pole je speciální struktura, která může v jedné proměnné obsahovat sadu hodnot. Hovoříme o tom, že pole má **prvky**; každý prvek má index neboli **klíč** a **hodnotu**.

Indexovaná pole

Příklad:

```
<?php
$arr = array(1, 2, 3, 4); // indexované pole
$arr2 = array(0=>1, 1=>2, 2=>3, 3=>4); // indexované pole
?>
```



Asociativní pole

Pole uvedené výše je **indexované**, protože jeho prvky jsou určeny indexem. Někdy může ale být šikovnější odlišit od sebe prvky pole řetězcem, čímž vznikne tzv. **asociativní pole**.

Příklad:

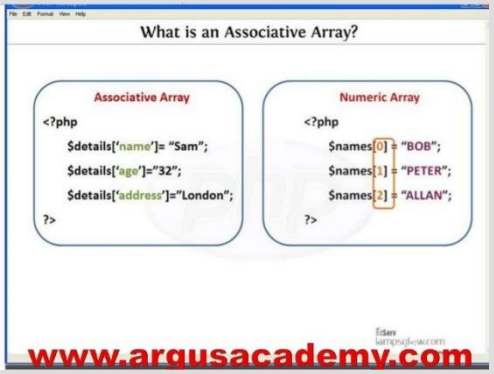
```
<?php
$obyvatel["Praha"]=1000000;
$obyvatel["Ústí nad Labem"]=100000;
$obyvatel["Horní Lhota"] = 350;
?>
```

Vícerozměrná pole

Pole obsahuje hodnoty; místo hodnoty ale můžeme mít opět pole.

Příklad:

```
<?php
$figura["a"][1]="bílá věž";
$figura["b"][1]="bílý jezdec";
//...atd...
$figura["h"][2]="bílý pěšec";
//...atd...
$figura["g"][8]="černý jezdec";
$figura["h"][8]="černá věž";
echo "Na poli b1 je při zahájení šachové partie ".$figura["b"][1];?>
```



Přednáška 2: Základy PHP

Pole (2/2)

Inicializace polí
Pole můžeme inicializovat, tzn. nastavit tak, že jeho prvkům **přiřadíme postupně hodnoty**. Navíc ovšem můžeme index pole vynechat a PHP jej **dosadí za nás**. Další možností je použít PHP funkci **array**, která provede totéž, ale je to mnohem kratší.
Příklad:

```
<?php
$arr[0] = 1; $arr[1] = 2; // inicializace přiřazením hodnoty
$arr[] = 1; $arr[] = 2; // PHP dosadí indexy (klíče)
$arr2 = array(0=>1, 1=>2, 2=>3, 3=>4); // inicializace pomocí funkce array
?>
```

Z hlediska významu pole mohou

- sloužit jako seznamy
- simulovat slovníky
- fungovat jako kolekce prvků
- pracovat jako zásobníky nebo fronty
- představovat stromové struktury (prvkem pole totiž může být pole)
- být vracena z funkcí PHP (například se to týká databází)
- s výhodou se dají použít při zpracování dat z formuláře (hranaté závorky za jménem prvku způsobí, že prvek se načte jako součást pole)