



PROGRAMOVANIE WEBOVÝCH STRÁNOK

~~ANDREA HRUŠECKÁ, ROMAN HRUŠECKÝ,~~
DANIELA BEZÁKOVÁ, ĽUDMILA JAŠKOVÁ



EURÓPSKA ÚNIA
Európsky sociálny fond
Európsky fond regionálneho rozvoja



OPERAČNÝ PROGRAM
ĽUDSKÉ ZDROJE



MINISTERSTVO
ŠKOLSTVA, VEDY,
VÝSKUMU A ŠPORTU
SLOVENSKEJ REPUBLIKY



Tento projekt sa realizuje vďaka podpore z Európskeho sociálneho fondu
a Európskeho fondu regionálneho rozvoja v rámci Operačného programu Ľudské zdroje

www.minedu.sk www.employment.gov.sk/sk/esf/ www.itakademia.sk

OBSAH

1	Dynamické webové stránky.....	4
1.1	Jazyk PHP, PHP stránky.....	5
1.2	Tvorba PHP stránok.....	8
1.3	Metodické pokyny pre učiteľa.....	11
2	PHP - Základná syntax a štruktúry (1).....	14
2.1	Príkaz echo, reťazce (1).....	14
2.2	Komentáre.....	16
2.3	Premenné.....	17
2.4	Konštanty.....	18
2.5	Reťazce (2).....	18
2.6	Operátory.....	22
2.7	Príkaz if.....	25
2.8	Cykly.....	27
2.9	Polia.....	29
2.10	Ladenie a analyzovanie chýb.....	33
2.11	Metodické pokyny pre učiteľa.....	37
3	PHP - spracovanie formulárov (1).....	40
3.1	Odoslanie formulára, tlačidlo.....	40
3.2	Textové pole - INPUT a TEXTAREA.....	41
3.3	Výberová ponuka - SELECT.....	44
3.4	Prepínač - RADIOBUTTON.....	45
3.5	Začiarkávacie políčko - CHECKBOX.....	46
3.6	Metodické pokyny pre učiteľa.....	49
4	PHP - Základná syntax a štruktúry (2).....	52

4.1	Náhodné čísla	52
4.2	Dátum a čas	52
4.3	Vlastné funkcie	55
4.4	Viaceré PHP vsuvky	60
4.5	Metodické pokyny pre učiteľa	69
5	PHP - spracovanie formulárov (2)	76
5.1	Zobrazenie odoslaných údajov znovu vo formulári	76
5.2	Skrývanie formulára	82
5.3	Kontrola vstupov od používateľa	85
5.4	Metodické pokyny pre učiteľa	96
6	PHP - vkladanie kódu z rôznych súborov	98
6.1	Pripojenie súboru - INCLUDE.....	98
6.2	Oddelenie obsahu stránky od funkcií.....	101
6.3	Metodické pokyny pre učiteľa	108
7	PHP - trvalé ukladanie údajov	110
7.1	Uloženie odoslaných údajov do súboru	110
7.2	JSON	112
7.2.1	Využitie JSON v PHP	112
7.3	Načítanie údajov zo súboru.....	113
7.4	Metodické pokyny pre učiteľa	116
8	PHP - pamätanie údajov.....	120
8.1	Metodické pokyny pre učiteľa	125

1 DYNAMICKÉ WEBOVÉ STRÁNKY

Vytvárať webové stránky len pomocou HTML, presnejšie písať celý obsah do zdrojového HTML súboru (tzv. statické stránky), nie je vždy to správne, resp. takým spôsobom niektoré stránky nevieme vytvoriť. Napríklad stránky pre internet banking, internetové obchody, rôzne diskusné fóra takýmto spôsobom vytvoriť nevieme, lebo nevieme aktuálny obsah. Ten sa totiž mení. Pri takýchto stránkach nám pomôže ich dynamická tvorba.

Rozdiel medzi statickými (HTML) a dynamickými stránkami je veľký. Pri statických stránkach sa na serveri nachádza finálna „verzia“ stránky, teda to, čo je uložené na serveri (súbor) sa presne zobrazí klientovi vo webovom prehliadači. **Dynamická stránka** však vzniká (generuje sa) až pri požiadavke klienta o jej zobrazenie. Vtedy sa vykonajú všetky potrebné činnosti a vygeneruje sa HTML kód výslednej stránky, ktorá sa zobrazí vo webovom prehliadači.

Dynamické stránky rozdeľujeme na 2 typy podľa toho, kde sa dynamická časť stránky generuje - či na serveri alebo v prehliadači. To, čo sa vykonáva v prehliadači, si vieme čiastočne pozrieť v zdrojovom kóde. Čo sa so stránkou deje na serveri, to nevieme zistiť. Na serveri sa totiž spustí kód programu a až jeho výsledok sa nám zobrazí v prehliadači a my nedokážeme zistiť, čo sa udialo - či sa vykonali nejaké funkcie, cykly, či sa vložili nejaké súbory atď. Toto všetko je pred nami (ako používateľmi) skryté.

Prečo tvoriť dynamické stránky? Povedzme, že by sme chceli:

- 1) vypísať dni v mesiaci (čísla 1 až 31) v rámci kalendára na stránke,
- 2) zobraziť na stránke nejakú akciu len v presne stanovenom období (dátum, čas),
- 3) vypísať zoznam mien, ktoré máme uložené napr. v poli/zozname alebo inej dátovej štruktúre,
- 4) vypísať zoznam článkov, ktoré niekto pripravil/napísal a sú uložené napr. v súbore alebo v poli/zozname,
- 5) vypísať zoznam najbližších predstavení nejakého kina na viacerých stránkach,
- 6) spracovať formulár (napr. komentovať, objednať si, rezervovať si, poslať správu správcovi stránky,...),
- 7) načítať údaje zo súboru alebo databázy, filtrovať ich a potom zobraziť na stránke,
- 8) uložiť údaje do súboru prípadne databázy.

Riešiť predchádzajúce problémy pomocou statickej webovej stránky by bolo neefektívne a vo väčšine prípadov dokonca nemožné. Napr. v prípade (1) by to znamenalo ručne vypísať do zdrojového kódu stránky 31 čísel – (tabuľka 1 vľavo), v prípade (2) nahráť stránku na server v presne stanovenom čase.

Pri riešení týchto úloh nám môžu pomôcť dynamické webové stránky - môžeme v nich využiť výpis čísel pomocou cyklu, výpis výsledkov výpočtov, výpis obsahu poľa/zoznamu prípadne súboru, načítanie údajov zo súboru alebo databázy, prípadne zápis údajov do súboru alebo databázy. Na tvorbu dynamických webových stránok môžeme využiť rôzne programovacie jazyky (napr. PHP, C#, ASP.NET, Java, JavaScript, Python...). Každý má svoje výhody aj nevýhody (aj fanúšikov a odporcov :-)).

Tabuľka 1.1 Príklad použitia PHP skriptu na generovanie položiek výberovej ponuky.

<pre><option value='1'>1</option> <option value='2'>2</option> <option value='3'>3</option> ... <option value='30'>30</option> <option value='31'>31</option></pre>	<pre>for (\$i = 1; \$i <= 31; \$i++) { echo "<option value='\$i'>\$i </option>\n"; }</pre>
Výpis čísel len v HTML	Výpis čísel pomocou PHP

V prvej časti učebnice sa budeme venovať tvorbe dynamických webových stránok na strane servera a stránky budeme vytvárať pomocou programovacieho jazyka **PHP**, preto ich budeme nazývať PHP stránky. PHP vie pracovať s rôznymi súbormi, komunikovať s rôznymi databázami, pracovať s poštovými službami. Tento jazyk je veľmi populárny, voľne dostupný, používajú ho aj veľké systémy, resp. spoločnosti ako napr. Facebook, Wikipédia, WordPress.

Druhá časť učebnice bude venovaná tvorbe dynamických webových stránok na strane klienta pomocou programovacieho jazyka **JavaScript**.

1.1 Jazyk PHP, PHP stránky

Program, ktorý spracováva tzv. HTTP požiadavky zvyčajne odoslaných z webového prehliadača. Po spracovaní požiadavky vráti webovému prehliadaču odpoveď, napr. webovú stránku, obrázok, čistý text...

K tvorbe a prezeraniu statických webových stránok nepotrebujeme okrem editora stránok a webového prehliadača nič iné. Naproti tomu, k zobrazeniu a testovaniu dynamických webových stránok na strane servera potrebujeme niekoľko vecí – webový server (**Apache**, IIS, ...), programovací jazyk na strane servera (**PHP**, ASP, Python, ...) a vo väčšine prípadov aj databázový server na ukladanie údajov (**MySQL**, MS SQL, SQLite, PostgreSQL, Oracle, ...). Dnešné webové aplikácie prakticky neexistujú bez databáz. Spolupráce PHP s databázami sa v našej učebnici dotkneme len okrajovo.

Čo sa deje s PHP stránkami?

Ako sa spracováva (zobrazuje) PHP stránka?

- 9) Používateľ zadá adresu PHP stránky do webového prehliadača (na lokálnom počítači).
- 10) Webový server (napr. Apache) rozpozná PHP a pošle PHP súbor tzv. PHP interpretu.
- 11) PHP interpret vykoná PHP kód v súbore a výsledok pošle späť webovému serveru.
- 12) Webový server pošle výsledok (výslednú stránku) webovému prehliadaču (na lokálnom počítači) a ten ju zobrazí.



Obrázok 1.1 Zobrazenie PHP stránky.

Stručne o jazyku PHP

PHP¹ je **skriptovací jazyk** s otvoreným kódom na strane servera. Skriptovací jazyk znamená, že program v ňom vytvorený sa nevykonáva samostatne, ale vykonáva ho tzv. interpretér (vykonávač). V prípade PHP je to PHP interpretér, ktorý musí byť súčasťou webového servera. Program v skriptovacom jazyku sa nazýva skript.

ZAPAMÄTAJTE SI

PHP súbory nemôžeme len tak otvoriť vo webovom prehliadači, ale musíme ich spustiť (vykonať) pomocou PHP interpretéra cez webový server.



Jazyk PHP je netyповý jazyk. V skutočnosti za každou premennou nejaký typ je (automaticky generovaný jazykom PHP), ale pri vytváraní premennej ho nemusíme nijako definovať.

Súbory s PHP skriptami majú zväčša koncovku `.php`.

Jazyk PHP je tzv. **CASE sensitive**, teda v ňom záleží na veľkosti písmen v názvoch premenných, konštánt, funkcií atď. V jazyku PHP sú teda príkazy `vypis()` a `Vypis()` rôzne príkazy.

Ako spúšťať PHP stránky na svojom počítači?

Ako sme už napísali vyššie, na spúšťanie (vykonávanie) PHP stránok potrebujeme viacero programov – webový server (napr. Apache), PHP interpretér a v prípade použitia databázy aj databázový server (napr. MySQL). Tie sa štandardne na počítačoch nenachádzajú, preto je nutné ich doinštalovať.

V súčasnosti už existujú aplikácie, ktoré v sebe integrujú všetky tri spomínané programy. Jednou z takých je program USB Webserver². USB Webserver netreba inštalovať, stačí ho rozbaľiť a spustiť súbor `usbwebserver.exe`. Tým sa spustí webový server spolu s PHP interpretérom.

Pri prvom spustení USB Webservera nastavíme jazyk, v ktorom sa bude zobrazovať program. Potom v časti *Settings* zmeníme hodnotu *Port apache* z 8080 na **80** a uložíme nastavenia kliknutím na tlačidlo *Save*. Webový server sa na pozadí reštartuje.

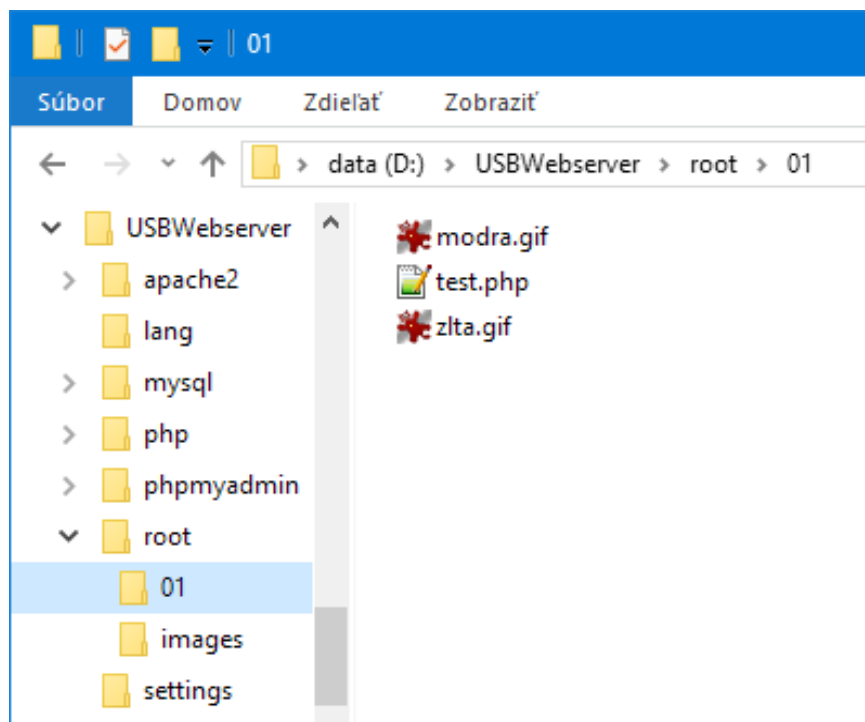
¹ Ďalšie informácie a aj manuály nájdete na stránkach www.php.net, www.php5.sk.

² <http://www.usbwebserver.net/webserver/>



Obrázok 2.2 Okno aplikácie USB Webserver - Settings.

PHP stránky nemôžeme ukladať na ľubovoľné miesto na počítači, ale len tam, kam „vidí“ webový server. Pri štandardnom nastavení USB Webservera je to priečink `root` v rámci priečinkov USB Webservera na počítači.



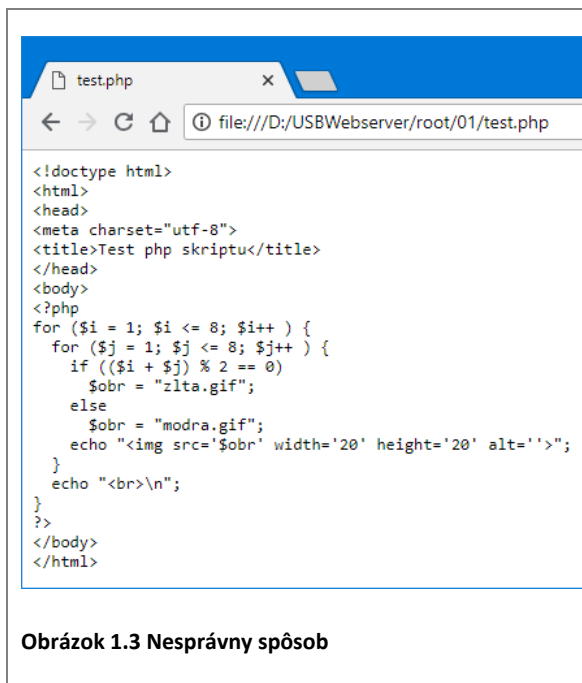
localhost
označujeme
webový server na
lokálnom počítači

Ak si chceme php stránky spustiť na svojom počítači, spustíme si ľubovoľný webový prehliadač a v ňom zadáme adresu `localhost`. Zobrazí sa úvodná stránka nášho webového servera.

Teraz si ukážeme správny a nesprávny spôsob spúšťania (testovania) PHP stránok.

Nesprávny spôsob spúšťania PHP stránok

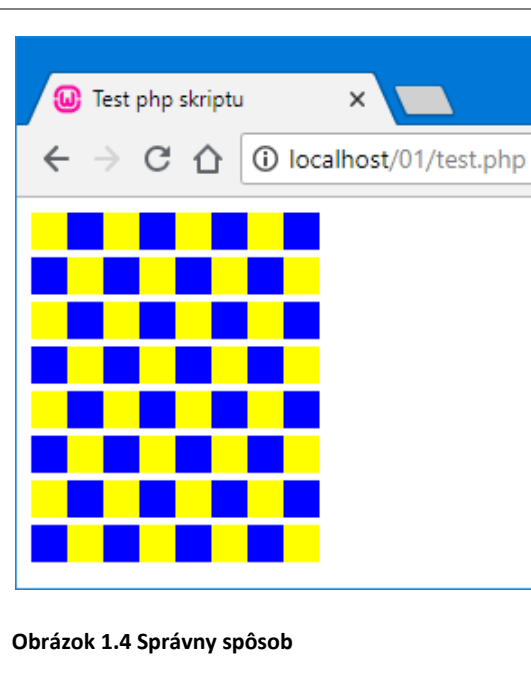
- Súbor -> Otvoriť... alebo
- Dvojkliknutie na PHP súbor



Vidíme PHP kód!

Správny spôsob spúšťania PHP stránok

- Zadanie adresy v prehliadači:
`http://localhost/01/test.php`



NEvidíme PHP kód, ale vidíme výslednú dynamicky vygenerovanú webovú stránku!

Pri niektorých nastaveniach prehliadača sa môže stať, že nesprávne otvorená PHP stránka sa ani nezobrazí, ale sa môže uložiť (pokúšať sa uložiť) na disk počítača.

1.2 Tvorba PHP stránok

Stránky s PHP kódom môžeme písať v ľubovoľnom editore webových stránok. Pomôže nám, ak editor zvýrazňuje PHP syntax (príkazy, zátvorky, štruktúry atď.). Najlepšie je používať taký editor, ktorý má v sebe integrované PHP funkcie (napr. Atom, Notepad++, PSPad, BlueFish, PhpStorm).

PHP stránka môže byť klasická HTML stránka s PHP vsuvkami, alebo môže byť celá zložená z PHP kódu.

ZAPAMÄTAJTE SI

PHP skript (kód) vkladáme medzi značky `<?php ... ?>` (túto časť nazývame PHP vsuvka). Do HTML kódu môžeme vložiť viacero PHP vsuviok.

Všetko, čo je v súbore mimo PHP vsuvky `<?php ... ?>`, sa automaticky prenesie do výsledného HTML súboru. Obsah PHP vsuvky sa vykoná a až jej výsledok sa vloží namiesto jej kódu. Výsledný zdrojový kód webovej stránky by mal byť korektný (validný) HTML kód.



V princípe môžeme PHP vsuvky vkladať na ľubovoľné miesto, ale musí byť zachovaná validita výsledného HTML kódu.



PRÍKLAD 1.1

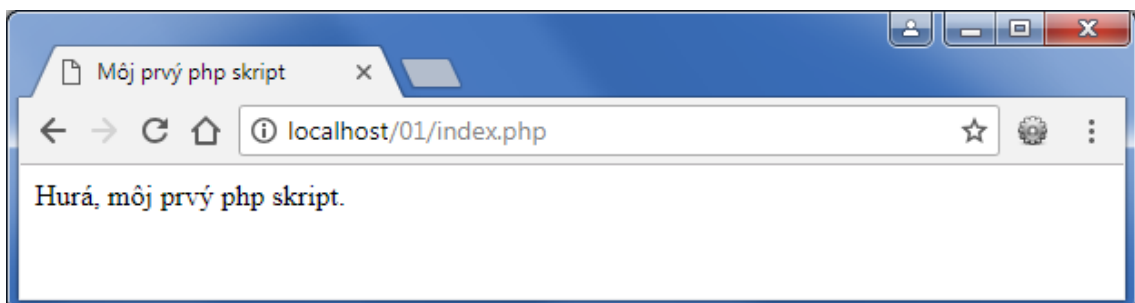
Vytváranie PHP kódu a generovanie výsledného zdrojového kódu webovej stránky si ukážeme na príklade. V nasledujúcom výpise je zdrojový kód stránky (súboru) `index.php` (`01/index.php`) spolu s PHP vsuvkou.

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Môj prvý php skript</title>
</head>
<body>
<?php echo "Hurá, môj prvý php skript."; ?>
</body>
</html>
```

Pri načítaní stránky do webového prehliadača (`http://localhost/01/index.php`) webový server na našom počítači vykoná pomocou PHP interpretéra PHP vsuvku a vygeneruje nový (a už výhradne HTML) zdrojový kód stránky:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Môj prvý php skript</title>
</head>
<body>
Hurá, môj prvý php skript.
</body>
</html>
```

Ten potom webový prehliadač interpretuje a zobrazí nám výslednú stránku ako na obrázku 1.5.



Obrázok 1.5 Zobrazenie súboru `index.php`

ZAPAMÄTAJTE SI

Pri korektnom zobrazení PHP stránky zo servera sa klient (používateľ cez webový prehliadač) nikdy nemôže dostať k zdrojovému PHP kódu, a teda nedokáže zistiť, ktorá časť stránky bola vygenerovaná PHP kódom a čo bol len statický HTML kód.



ÚLOHA 1.2

Otvorte súbor `prvy.html` (z priečinku 01). Premenujte ho na `prvy.php`. Obsah súboru bude tvoriť zdrojový kód HTML stránky spĺňajúci štandard HTML5.

- Do tela stránky (element `<body>`) vložte PHP vsuvku s príkazom
`echo "Toto je môj prvý PHP skript. Hurá, funguje.";`

Vyskúšajte si váš súbor zobraziť (spustiť) vo webovom prehliadači. Nezabudnite si pozrieť aj zdrojový kód zobrazenej stránky v prehliadači. Dbajte na to, aby výsledná stránka bola validným HTML dokumentom.



1.3 Metodické pokyny pre učiteľa



CIEĽ

Cieľom tejto kapitoly je priblížiť tvorbu dynamických webových stránok na strane servera. Študenti sa dozvedia rozdiel medzi statickými a dynamickými webovými stránkami, oboznámia sa s nástrojmi na ich tvorbu, s tým, ako spúšťať dynamické stránky na strane servera a vytvoria prvý veľmi jednoduchý PHP súbor.



VÝKLAD

Prvá časť je hlavne teoretická. Študentov treba oboznámiť s tým:

- čo sú dynamické webové stránky,
- čo všetko je potrebné pri tvorbe dynamických webových stránok na strane servera,
- ako sa spracovávajú PHP stránky, keď príde požiadavka od klienta (prehliadača),
- ako sa do stránok vkladá PHP kód.

Pre prácu odporúčame používať USB Webserver³, čo je kombinácia populárnych webserverových aplikácií – Apache, PHP, MySQL a phpMyAdmin. Webserver netreba inštalovať, stačí ho rozbaľiť a spustiť `usbwebserver.exe`.



Obrázok 1.6 Spustený USB Webserver

Na začiatku hodiny alebo pred samotným testovaním PHP stránok treba spustiť USB Webserver. Stránky treba nakopírovať do priečinka `root` v priečinku `USBWebserver`. Keď sa zadá v prehliadači `localhost`, tak sa zobrazí práve obsah priečinka `root`.

³ <http://www.usbwebserver.net/webserver/>

Ak na počítači pracujú (postupne) viacerí študenti, odporúčame vytvoriť pre nich v priečinku `root` podpriečinky označené ich menami. Prípadne v nastaveniach USB Webservera treba zmeniť domovský priečinok na nejaký zdieľaný priečinok na školskom serveri.



Ukážkové súbory a aj súbory potrebné na vyriešenie úloh je potrebné nakopírovať do pracovných priečinkov každého študenta alebo do nejakého zdieľaného školského priečinka.

Okrem chybových správ v súbore `error.log` (pre USB Webserver je v priečinku `apache/logs`) sa priebežne budeme zaoberať chybovými správami, ktoré sa zobrazujú aj priamo v prehliadači.

[Ku časti: Ako skúšať PHP stránky na svojom počítači?](#)

Pri nesprávnom zobrazení PHP stránky prehliadače Edge a Firefox zobrazia prázdnu stránku (ak sa vypisuje jednoduchý text) alebo nejaký kus kódu. Vtedy treba pozrieť zdrojový kód stránky v prehliadači. Toto treba študentom stále zdôrazňovať - pozerajte si zdrojový kód stránky v prehliadači.

[K webovému serveru:](#)

Namiesto programu USB Webserver môžeme využiť iné programy/balíky napr. WAMP⁴, XAMPP⁵, EasyPHP⁶.

⁴ Windows-Apache-MySQL-PHP, www.wampserver.com

⁵ XAMPP, www.apachefriends.org

⁶ EasyPHP, www.easyphp.org

2 PHP - ZÁKLADNÁ SYNTAX A ŠTRUKTÚRY (1)

V tejto časti sa oboznámime s väčšinou základných štruktúr jazyka PHP. Veríme, že väčšinu pojmov poznáte z iných programovacích jazykov, takže si treba hlavne všímať, ako jednotlivé prvky zapisujeme v PHP.

Príkazy v jazyku PHP oddeľujeme bodkočiarkou `;`.

Ak nebude v texte špeciálne uvedené, budeme príkazy zadávať do PHP vsuvky v hlavnom obsahu stránky (do elementu `<body>`).

2.1 Príkaz echo, reťazce (1)

Ak chceme niečo vypísať do výsledného HTML súboru, použijeme príkaz `echo`. Pomocou neho môžeme vypisovať čísla (celé, reálne) a texty.

PRÍKLAD 2.1

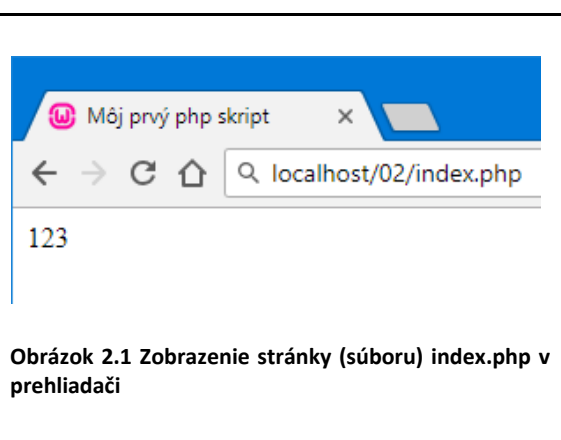
Vypíšme do tela stránky (elementu `body`) číslo 123.



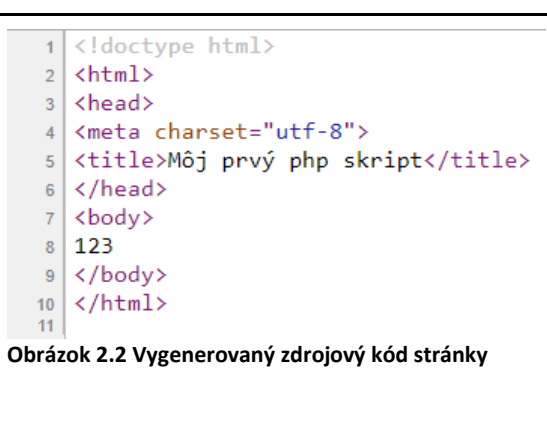
02/index01.php

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Môj prvý php skript</title>
</head>
<body>
<?php echo 123; ?>
</body>
</html>
```

Zdrojový kód uložíme do súboru `index.php` a otvoríme vo webovom prehliadači. Nezabudnite, že v prehliadači zadáme adresu `localhost/02/index.php`.



Obrázok 2.1 Zobrazenie stránky (súboru) `index.php` v prehliadači



Obrázok 2.2 Vygenerovaný zdrojový kód stránky



ÚLOHA 2.2

Vyskúšajte zadávať nasledujúce príkazy a zistite, čo sa zobrazí v prehliadači (príkazy musíte vložiť do PHP vsuvky).

- `echo 3.14;` (desatinnú čiarku píšeme ako bodku `.`)
- `echo "Hurá, môj prvý php skript.";`
- `echo 'Hurá, môj prvý php skript.';`

Texty (reťazce) v PHP musíme vložiť do apostrofov (`'reťazec1'`) alebo úvodzoviek (`"reťazec2"`). Takto definované reťazce však nie sú rovnaké, resp. PHP k nim nepristupuje rovnako. Rozdiely si vysvetlíme neskôr. Pre bežné texty je jedno, ktorý typ reťazca použijeme.

Ak vypisujeme texty, v skutočnosti vytvárame (generujeme) HTML zdrojový kód. Teda do reťazcov nemusíme dať len čistý text, ale môžeme v nich použiť aj HTML elementy.



PRÍKLAD 2.3

02/index03.php

Vypíšme reťazce oboma spôsobmi (pomocou úvodzoviek aj apostrofov), ale každý do samostatného riadka.

```
<body>
<?php
echo "Hurá, môj prvý php skript.<br>";
echo 'Hurá, môj prvý php skript.';
?>
</body>
```

	<pre>1 <!DOCTYPE html> 2 <html> 3 <head> 4 <meta charset="utf-8"> 5 <title>Môj prvý php skript</title> 6 </head> 7 <body> 8 Hurá, môj prvý php skript.
Hurá, môj prvý php skript. 9 </body> 10 </html></pre>
--	---



ÚLOHA 2.4

Vypíšte texty: $3 + 4$, $3 * 4$, prípadne aj ďalšie príklady. Skúste ich vypísať pomocou jedného príkazu `echo`. A potom ich vypísať pomocou viacerých príkazov `echo`. Snažte sa, aby príklady boli v prehliadači pod sebou.

PRÍKLAD 2.5



Vypíšme príklady z úlohy 2.4 tak, že každý príklad bude v samostatnom odseku.

02/index05.php

```
<?php
echo '<p>3 + 4</p>';
echo '<p>3 * 4</p>';
?>
```

```
1 <!doctype html>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <title>Môj prvý php skript</title>
6 </head>
7 <body>
8 <p>3 + 4</p><p>3 * 4</p>
9 </body>
10 </html>
```

ÚLOHA 2.6



Vypíšte hlavný nadpis stránky (`h1`) s textom *Nadpis PHP stránky* a pod ním odsek s textom *Odsek textu s príkladom: 3 * 4* ako na obrázku 2.3.

Nadpis PHP stránky

Odsek textu s príkladom: 3 * 4

Obrázok 2.3 Nadpis a odsek

2.2 Komentáre

Niekedy si pri písaní kódu potrebujeme niečo „poznačiť“, prípadne doplniť komentár ku bloku kódu, teda texty, ktoré sa nemajú vykonávať. V PHP môžeme vložiť dva typy komentárov.

- 1) `//` – takýto komentár platí po koniec riadka alebo po koniec PHP vsuvky (značka `?>`)
- 2) `/*` `viacriadkový komentár` `*/` – toto je komentár, ktorý môžeme vložiť kdekoľvek v kóde a môže zasahovať do viacerých riadkov.


```

7 <body>
8 <?php
9 // Príklad 2.3 - výpis elementu
10 echo "Hurá, môj prvý php skript.<br>";
11 echo 'Hurá, môj prvý php skript.';
12
13 /* Úloha 2.4
14 echo "3 * 4";
15 echo "3 + 4";
16 echo "3 - 4";
17 echo "3 / 4";
18 */
19 ?>
20 </body>

```



ZAPAMÄTAJTE SI

Nesmieme zabúdať na to, že takéto komentáre platia iba v rámci PHP vsuvky `<?php ... ?>`

2.3 Premenné

Každá premenná musí začínať znakom `$`. Tým sa odlišuje od ostatných štruktúr a textov, napr. `$pocet`, `$meno`.

V názve premennej môžu byť len písmená, číslice a podčiarkovník (`_`). Hneď za znakom `$` nesmie nasledovať číslica.

Premennú v PHP vytvárame **priradením** hodnoty do premennej, teda napr. príkazom `$cislo = 123`. Nie je nutné premenné nejako špeciálne definovať ani určiť typ premenných. PHP si vnútorne definuje typ premennej podľa toho, akú jej priradíme hodnotu.



PRÍKLAD 2.7

Zadefinujme si niekoľko premenných s rôznymi hodnotami a vypíšme ich.

```

$cislo = 123;           //(typ celé číslo)
$realne = 12.3;       //(typ reálne číslo)
$ret1 = "Ahoj";       //(typ reťazec)
$ret2 = 'Ahoj';       //(typ reťazec)
echo $cislo;
echo $ret1;

```

02/index07.php

2.4 Konštanty

Konštanta je ako keby premenná, ktorá má stále tú istú hodnotu a v kóde ju už nemôžeme zmeniť.

Ku konštantám sa musíme správať inak ako k premenným. Skôr ako chceme používať nejakú konštantu, musíme ju definovať. Konštantu definujeme príkazom `define("názov_konštanty", hodnota)`.

Názvy konštant nesmú začínať znakom `$` a tiež nesmú začínať číslom. Zvyčajne ich píšeme veľkými písmenami (aby sme ich odlišili od textov a premenných), ale nie je to žiadne striktné pravidlo.

PRÍKLAD 2.8

Definujme konštantu `POCET` s hodnotou `12` a vypíšme ju. Potom definujme konštantu `AUTOR` s hodnotou `Janko Hraško` a tiež ju vypíšme.

```
define("POCET", 12);  
echo POCET;  
echo '<br>';  
define("AUTOR", 'Janko Hraško');  
echo AUTOR;
```



2.5 Reťazce (2)

V predchádzajúcich príkladoch sme si vytvorili rôzne premenné a konštanty a vypisovali sme ich. Zvyčajne ich však vypisujeme spolu s nejakým textom. Napríklad:

```
$cislo = 123;  
echo "Výsledok je ";  
echo $cislo;
```

Takýto zápis je korektný, ale zbytočne predlžuje kód. V časti 2.1 (Reťazce (1), príkaz echo) sme písali, že reťazce môžeme zapisovať do úvodzoviek aj apostrofov. Poďme si ukázať rozdiely a takisto, ako nám môžu skrátiť a sprehľadniť kód.

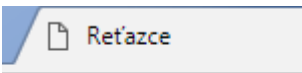
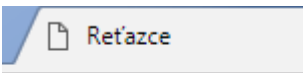
Reťazec **v úvodzovkách** PHP interpreter analyzuje a v prípade, že v ňom identifikuje (nájde) premenné alebo špeciálne znaky, vyhodnotí ich. Naproti tomu v reťazci **v apostrofoch** PHP interpreter nič neanalyzuje a celý reťazec vypíše, ako je – premenné a špeciálne znaky sa vypíšu ako obyčajné znaky.

PRÍKLAD 2.9

V nasledujúcich skriptoch sa pokúsime vypísať výsledok (premennú) spolu s textom. Jediný rozdiel je v zápise (definovaní) reťazca.



02/index09.php

<pre>\$cislo = 123; echo "Výsledok je \$cislo";</pre>	<pre>\$cislo = 123; echo 'Výsledok je \$cislo';</pre>
 Výsledok je 123	 Výsledok je Scislo

Vidíme, že kým v príkaze, uvedenom v ľavej časti tabuľky, PHP interpret analyzoval reťazec a našiel premennú (a aj ju vyhodnotil), tak v príkaze, uvedenom v pravej časti tabuľky, sa vypísal reťazec, ako sme ho zadali.



02/index10.php

PRÍKLAD 2.10

Zmeňme výpis z predchádzajúceho príkladu tak, aby hodnota čísla bola zvýraznená tučným písmom.

<pre>\$cislo = 123; echo "Výsledok je \$cislo";</pre>	
--	--



ÚLOHA 2.11

Porozmýšľajte a potom overte, čo vypíšu nasledujúce príkazy:

```
$cislo = 20;
echo "Môj brat má $cislo rokov.<br>";
echo '$cislo je moje šťastné číslo.<br>';
echo "<p>Vyrieš príklad $cislo - 4</p>";
echo 'Premenná <em>$cislo</em> má hodnotu $cislo<br>';
echo "Premenná <em>$cislo</em> má hodnotu $cislo<br>";
```

Zreťazovanie

V predchádzajúcich častiach sme vypisovali samostatné texty, premenné, konštanty alebo texty v kombinácii s premennými. Skúsme teraz jedným príkazom `echo` vypísať text a konštantu. Ako prvé by nás zrejme napadlo toto riešenie:

```
define("POCET", 12);
echo "Počet záznamov je: POCET";
```

Keď však toto riešenie vyskúšame, zistíme, že namiesto čísla 12 (hodnoty konštanty `POCET`) sa vypíše text `POCET` a to napriek tomu, že sme na výpis použili úvodzovky. Keďže názvy konštánt sa nezačínajú žiadnym špeciálnym znakom, PHP ich v reťazci nemá ako identifikovať. Konštanty teda na rozdiel od premenných **nemôžeme** vypisovať priamo v reťazci. Môžeme však využiť

zreťazovanie (spájanie) reťazcov, ktoré robíme pomocou znaku bodka (.). Upravme predchádzajúci kód nasledovne:

```
define("POCET", 12);  
echo "Počet záznamov je: " . POCET;
```

V tomto prípade už konštanta POCET nie je súčasťou reťazca uzavretého v úvodzovkách, ale samostatnou časťou, ktorú pomocou znaku . pripájame za text "Počet záznamov je: ". Druhý riadok kódu by sme mohli interpretovať takto: *Vypíš reťazec, ktorý dostaneš spojením textu "Počet záznamov je: " a hodnoty konštanty POCET.* Teraz teda dostaneme očakávaný výpis: *Počet záznamov je: 12*

ÚLOHA 2.12

Pozrite a vyskúšajte si aj nasledujúce príklady zreťazenia. Potom zmeňte hodnoty premenných \$cislo a \$retazec, či konštanty POCET a skúste znova.

```
$cislo = 10;  
define("POCET", 12);  
$retazec = "PHP";  
echo "Konštanta POCET má hodnotu " . POCET . "<br>";  
echo "Na rukách máme " . $cislo . " prstov.<br>";  
echo $retazec . " je super.<br>";  
echo POCET . $retazec . $cislo . "<br>";  
$veta = POCET . "-krát napíš: <em>Mám rád " . $retazec .  
"!</em><br>";  
echo $veta;
```

Vidíme, že PHP zo všetkých častí zreťazenia spraví jeden reťazec. Ten následne môžeme napr. vypísať, alebo ho priradiť do premennej. Nemusíme riešiť to, či jednotlivé časti sú text alebo číslo – PHP ich bez problémov spojí.

ÚLOHA 2.13

Pomocou PHP kódu vypíšte text: `Premenná $cislo má hodnotu 10.`, pričom posledné číslo vo vete bude skutočná hodnota premennej \$cislo.

Vypisovanie špeciálnych znakov

Pozrime si zdrojový kód stránky, ktorú sme vytvorili v príklade 2.12 (pozor, nie PHP kód, ale HTML kód už vygenerovanej stránky).

```
7 <body>  
8 Konštanta POCET má hodnotu 12Na rukách máme 10 prstov.<br>PHPje super.<br>12PHP1012-krát napíš:  
9 </body>
```

Vidíme, že všetky texty, ktoré sme vypisovali príkazom `echo`, sa vypísali vedľa seba. Vytvorili sme tak veľmi zle čitateľný HTML kód, v ktorom sa ťažko orientuje a zle sa v ňom hľadajú chyby. Ak by sme chceli vytvoriť krajší html kód, mali by sme zabezpečiť, aby každý príkaz `echo` "zapísal na nový riadok". V tomto prípade nám nepomôže vložiť element `
`, lebo ten **nezalamuje** riadok v zdrojovom kóde, ale v samotnom prehliadači.



02/index12.php



Zalomenie riadka v zdrojovom HTML kóde riešime pomocou špeciálneho znaku `\n`, ktorý **musíme** vypisovať v úvodzovkách.



02/index14.p
hp

PRÍKLAD 2.14

Doplňte zalomenie riadka `\n` do zdrojového kódu stránky.

```
echo "<h1>Nadpis stránky</h1>\n";  
echo '<p>Odsek textu</p>';  
echo "<h1>Nadpis stránky</h1>\n<p>Odsek textu</p>";  
echo "<h2>Podnadpis</h2>\n<p>1. odsek</p>\n<p>2. odsek</p>";
```

HTML kód, ktorý vytvoríme týmto skriptom, vyzerá nasledovne:

```
7 <body>  
8 <h1>Nadpis stránky</h1>  
9 <p>Odsek textu</p><h1>Nadpis stránky</h1>  
10 <p>Odsek textu</p><h2>Podnadpis</h2>  
11 <p>1. odsek</p>  
12 <p>2. odsek</p>  
13 </body>
```



ÚLOHA 2.15

Upravte PHP skript z príkladu 2.12 tak, aby generoval čitateľný HTML kód.

Úvodzovky a apostrofy v reťazcoch

Pri vypisovaní znakov apostrof (') a úvodzovky (") si musíme dať veľký pozor. Nemôžeme ich len tak vložiť do reťazca, ktorý začína tým istým znakom, lebo by to PHP interpretér pochopil ako ukončenie reťazca a nevedel by, čo so zvyšným textom.

Nesprávny zápis

```
echo "text "xyz" text";
```

```
echo 'text 'xyz' text';
```

Správny zápis

```
echo "text 'xyz' text";
```

```
echo 'text "xyz" text';
```

Pre výpis úvodzoviek alebo apostrofov v texte môžeme tiež využiť znak `\`, ktorý zruší význam nasledujúceho znaku.

```
echo "text \"xyz\" text";
```

```
echo 'text \'xyz\' text';
```



ÚLOHA 2.16

Máme definované premenné `$obrazok = "obrazky/logo.jpg"` a `$adresa = "www.rtvs.sk"`. Pomocou PHP (príkazu `echo`) vytvorte HTML kód, v ktorom bude:

- element `Facebook`,
- element `<img src='obrazky/ikona.gif' alt='ikona' width='20'`

```
height='20'>
```

- element, definujúci odkaz na stránku, ktorej adresa je uložená v premennej `$adresa` (textom odkazu môže byť tiež hodnota premennej `$adresa`),
- element definujúci obrázok, ktorého adresa je uložená v premennej `$obrazok` vrátane atribútu `alt`). Upravte PHP skript z príkladu 2.12 tak, aby generoval čitateľný HTML kód.

2.6 Operátory

V PHP rozlišujeme niekoľko typov operátorov. Najdôležitejšie z nich si vysvetlíme a ukážeme.

Číselné

Základné operátory sčítanie, odčítanie a násobenie (+, -, *) netreba ani predstavovať. Fungujú tak, ako by sme očakávali.

Operátor delenia / funguje na celých a desatinných číslach, ale jeho výsledok je **vždy** desatinné číslo.

Operátor % (zvyšok po delení celých čísel) funguje len na celých číslach. V prípade, že ho použijeme pri desatinných číslach, tak PHP odtrhne desatinné časti a vykoná operáciu so zvyšnými celými číslami. Napr.

- príkaz `echo 10 % 3;` vypíše `1`,
- aj príkaz `echo 10.5 % 3;` vypíše `1`,
- aj príkaz `echo 10.5 % 3.7;` vypíše `1`,
- príkaz `$zvysok = 12 % 4;` priradí do premennej `$zvysok` hodnotu `0`.

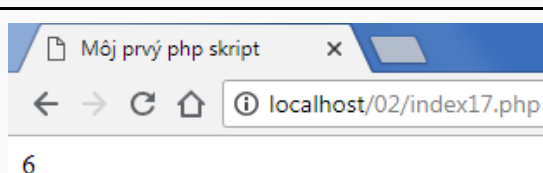
Jazyk PHP poskytuje tiež operátory zvýšenia hodnoty o 1 (++) a zníženia hodnoty o 1 (--). Sú to vlastne skrátene zápisy operácií + 1 a - 1. Napríklad zápis `$vysl++` by sme mohli zapísať aj ako `$vysl = $vysl + 1`. Podobne je to aj pri operátore --.

Operátory zvýšenia a zníženia hodnoty o 1 spájame len s premennými a medzi premennou a operátorom nesmie byť medzera.

PRÍKLAD 2.17

Vyskúšajme si operátor zvýšenia hodnoty premennej `$vysl` o 1. Výsledok kódu bude číslo 6.

```
$vysl = 5;  
$vysl++;  
echo $vysl;
```



02/index17.php

Priradovacie

Okrem základného operátora priradenia (=) v PHP existujú aj tzv. zložené priradovacie operátory, ktoré sú kombináciou dvoch operátorov, teda sú to skrátene zápisy.

`+=`, `-=`, `*=`, `/=`, `%=`, `.=`

Napríklad zápis `$vysl += 4` je v skutočnosti príkaz `$vysl = $vysl + 4`. Oba sú však ekvivalentné a záleží len na používateľovi, ktorý bude používať. (Takéto priradenia sú aj v jazyku Python.)



02/index18.php

PRÍKLAD 2.18

Vyskúšajme si operátor zloženého priradenia. Výsledkom bude číslo 10.

<pre>\$vysl = 5; \$vysl *= 2; echo \$vysl;</pre>	
--	--



ÚLOHA 2.19

Máme kód:

```
$vysl = 12;
$vysl ..... ;
echo $vysl;
```

Aký **zložený** priradovací príkaz treba doplniť namiesto bodiek a s akým číslom, aby sa v prehliadači vypísalo číslo: a) 24, b) 29, c) 3, d) 5? Má každá z týchto úloh len jediné riešenie?

Existuje aj operátor `and` – a zároveň (môžeme ho písať aj veľkými písmenami – `AND`) a operátor `or` – alebo (môžeme ho písať aj veľkými písmenami – `OR`). Avšak tieto operátory sa nemusia vyhodnocovať v rovnakom poradí ako operátory `&&` a `||`, preto ich nebudeme používať.

Logické

Medzi základné logické operátory patria:

- `&&` – a zároveň,
- `||` – alebo,
- `!` – negácia (ak použijeme operátor spolu s premennou, tak medzi operátorom a názvom premennej nesmie byť medzera, napr. `!$vysledok`).

Pri vytváraní logických výrazov môžeme používať zátvorky.



PRÍKLAD 2.20

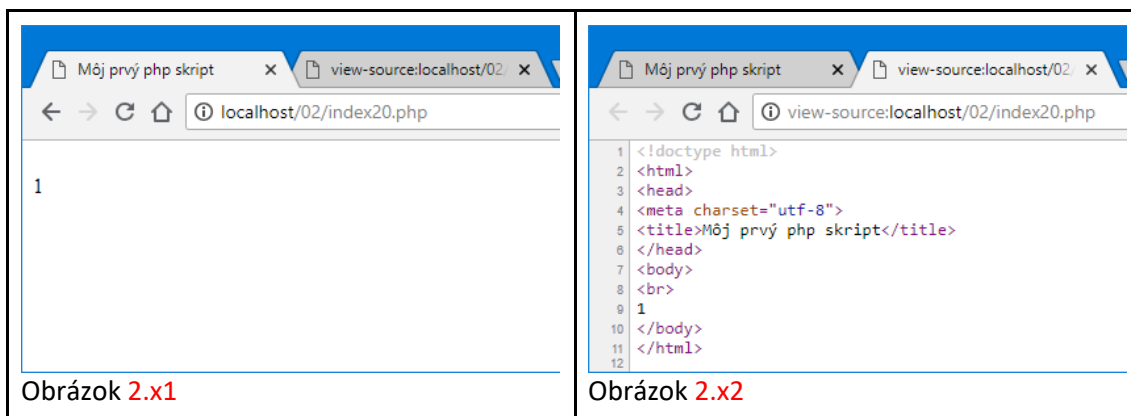
Čo bude výsledkom nasledujúcich príkazov?

```
$a = true; $b = false;
$vysledok1 = $a && $b;
$vysledok2 = $a && !$b;
```

Dajme si jednotlivé výsledky vypísať (každý v samostatnom riadku):

```
echo $vysledok1;
echo "<br>\n";
echo $vysledok2;
```

Zistíme, že v prehliadači máme jeden prázdny riadok, teda sa nám nevy písala premenná `$vysledok1` (pozri obrázok 2.x1, resp. zdrojový kód na obrázku 2.x2). V skutočnosti sa premenná `$vysledok1` vypísala, ale jej hodnota bol prázdny reťazec, lebo výsledok operácie `&&` bola hodnota `FALSE`, ktorú PHP pri vypisovaní prerobil na prázdny reťazec. Naproti tomu, v premennej `$vysledok2` mala byť hodnota `TRUE`, ale PHP vypísalo `1`, teda zase PHP pri vypisovaní prerobil `TRUE` na číslo `1`.



Z uvedeného príkladu môžeme vidieť, že PHP automaticky konvertuje typy hodnôt podľa potreby a situácie. Ak by sme hodnoty premenných `$a` a `$b` zmenili na číselné, výsledok by bol rovnaký. Pri premennej `$a` je jedno aké veľké číslo by sme dali.

```
$a = 1; $b = 0; // $a = 123;
$vysledok1 = $a && $b;
```

ZAPAMÄTAJTE SI

Logickej hodnote `FALSE` sú ekvivalentné hodnoty `0`, `0.0`, prázdny reťazec, `"0"`, prázdne pole a špeciálna hodnota `NULL`. Všetky ostatné hodnoty chápe PHP ako `TRUE`.

ÚLOHA 2.21

Zistite, aká hodnota bude v premennej `$vysledok` po jednotlivých priradeniach.

```
$c = 1; $d = 0; $e = 0;
$vysledok1 = $c || $d && $e;
$vysledok2 = ($c || $d) && $e;
```

Porovnávanie

Medzi základné operátory porovnania patria: `==`, `!=` (nerovnosť), `<`, `>`, `<=`, `>=`.


```
$vysledok = 3 < 5;
echo $vysledok;
echo 3 > 5;
```

2.7 Príkaz if

Podmienny príkaz `if` použijeme vtedy, ak chceme časť kódu vykonať len pri splnení nejakej podmienky (prípadne viacerých podmienok). Základná verzia príkazu je:

```
if (podmienka) {
    // príkazy
}
```

V časti `podmienka` zapíšeme jednu alebo viacero podmienok, ktoré chceme otestovať. Ak používame viacero podmienok, spájame ich logickými operátormi. Podmienka musí byť vždy v zátvorkách `()`.



02/podmienk
y22.php

PRÍKLAD 2.22

Naprogramujme dva podmienené výpočty:

- Do premennej `$body` uložíme počet bodov získaných za test. Ak študent získa viac ako 60 bodov, vypíšeme text *Test si spravil*.
- Do premennej `$den` uložíme nejaký deň v týždni. Ak to bude sobota alebo nedeľa vypíšeme text *Krásny víkend*.

Skúšajme meniť hodnoty premenných `$body` a `$den`.

```
$body = 85;
if ($body > 60) {
    echo 'Test si spravil';
}
echo '<br>';
$den = 'sobota';
if ($den == 'sobota' || $den == 'nedela') {
    echo 'Krásny víkend!';
}
```

Ak chceme vykonať nejaký príkaz (príkazy) vtedy, keď podmienka neplatí, môžeme použiť zápis príkazu `if` s časťou `else`. Časť `else` môžeme použiť len raz.

```
if (podmienka) {
    // príkazy 1
} else {
    // príkazy 2
}
```



02/podmienk
y23.php

PRÍKLAD 2.23

Naprogramujme dva podmienené výpočty:

- Do premennej `$body` uložíme počet bodov získaných za test. Ak študent získa viac ako 60 bodov, vypíšeme text *Test si spravil*.
- Do premennej `$den` uložíme nejaký deň v týždni. Ak to bude sobota alebo nedeľa vypíšeme text *Krásny víkend*.

PRÍKLAD 2.24

Takúto podmienku môžeme využiť napr. pri vypisovaní známky na základe bodov.

```
$body = 85;
if ($body > 90) {
    echo 1;
} elseif ($body > 80) {
    echo 2;
} elseif ($body > 70) {
    echo 3;
} elseif ($body > 60) {
    echo 4;
} else {
    echo 5;
}
```



02/podmienky2
4.php

ÚLOHA 2.25

Napište skript, ktorý podľa hodnoty premennej `$cas` vypíše jednu zo správ:

- *Dobré ráno*, ak je menej ako 8 hodín,
- *Dobrý deň*, ak je viac ako 8 hodín, ale menej ako 18,
- *Dobrý večer*, ak je medzi 18 a 22 a
- *Dobrá noc* v ostatných prípadoch.

Svoj skript vyskúšajte pre rôzne hodnoty premennej `$cas` (len medzi 0 a 24).



PRÍKLAD 2.26

Zoberme kód z príkladu 2.20:

```
$a = 1; $b = 0;
$vysledok1 = $a && $b;
```

Ak by sme nechceli premennú `$vysledok1` rovno vypisovať, ale podľa nej vypísať príslušné texty, mohli by sme zapísať



```
$a = 1; $b = 0;
$vysledok1 = $a && $b;
if ($vysledok1) {
    echo 'platí';
} else {
    echo 'neplatí';
}
```

Vidíme, že do podmienky nemusíme dať vždy skutočnú podmienku (výraz), ale môžeme dať len premennú. Ak v premennej nie je logická hodnota (**TRUE/FALSE**), PHP aplikuje automatické konvertovanie, ktoré sme uviedli v časti *Logické operátory*. Napr. ak by sme do podmienky vložili premennú s nenulovým číslom, PHP ju vyhodnotí ako **TRUE**. Môžeme to využiť napr. vtedy, ak chceme ďalej pracovať len s nenulovou alebo neprázdnu premennou.



ZAPAMÄTAJTE SI

Podmienka musí byť vždy v zátvorkách ().

2.8 Cykly

Ak potrebujeme viackrát vykonať rovnaké príkazy, môžeme použiť niektorý cyklus (napr. **for** alebo **while**).

Cyklus for

Cyklus **for** zapisujeme nasledovne:

```
for (inicializácia; podmienka; príkaz) {
    // príkazy
}
```

- **inicializácia** – príkaz v tejto časti sa vykonáva iba na začiatku cyklu (zvyčajne inicializácia hodnoty počítadla cyklu)
- **podmienka** – táto časť je zvyčajne výraz – ak tento výraz je **true**, zoznam príkazov vnútri cyklu sa vykoná, inak sa cyklus ukončí
- **príkaz** – príkaz v tejto časti sa vykoná na konci každého cyklu (zvyčajne zvyšovanie počítadla cyklu)



PRÍKLAD 2.27

Vypíšme pomocou cyklu **for**

- čísla od 1 do 10
- násobky čísla 5 v opačnom poradí od 100 do 0, teda 100 95 90 5 0

Mohli by sme to vyriešiť aj inak?

02/cykly27.p
hp

```

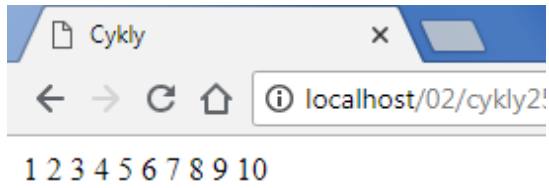
for ($i = 1; $i <= 10; $i++) {
    echo "$i ";
}

for ($i = 100; $i >= 0; $i = $i - 5) {
    echo "$i ";
}

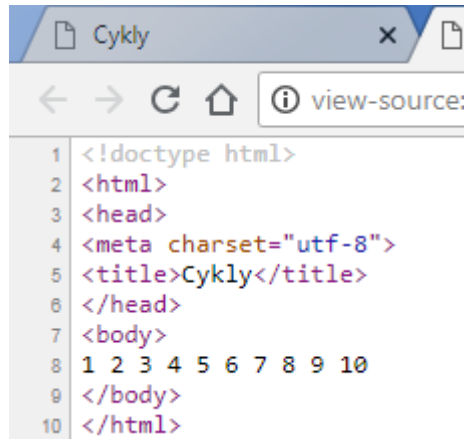
```

Zobrazenie prvého cyklu (a) v prehliadači a v zdrojovom kóde:

zobrazenie v prehliadači



zdrojový kód



ÚLOHA 2.28

Pomocou cyklu vypíšte:

- násobky čísla 5 do 100, teda 0 5 10 15 ... 95 100,
- prvých 10 čísel Fibonacchiho postupnosti, teda 1 1 2 3 5 8 13 21 34 55.



Ďalšie číslo postupnosti vypočítame ako súčet posledných dvoch čísel postupnosti. Prvé a druhé číslo postupnosti sú 1.

Cyklus while

Druhým typom cyklu je cyklus `while`. Ten zapisujeme nasledovne:

```

while (podmienka) {
    // príkazy
}

```

Ak podmienka platí, vykonávajú sa príkazy v tele cyklu.. Podmienka musí byť vždy v zátvorkách ().

PRÍKLAD 2.29

Vypíšme pomocou cyklu `while`

- čísla od 1 do 10
- násobky čísla 5 od 0 do 100



02/cykly29.php

```

$i = 1;
while ($i <= 10) {
    echo "$i ";
    $i++;
}
$i = 0;

while ($i <= 100) {
    echo "$i ";
    $i = $i + 5; // $i += 5;
}

```



ÚLOHA 2.30

Napište skript, ktorý vypíše:

- do jedného riadku mocniny dvojky menšie ako hodnota premennej `$koniec`; napr. ak `$koniec` je 100, bude výpis nasledovný: 1 2 4 8 16 32 64
- do jedného riadku čísla od 0 do 30, pričom násobky trojky zvýrazní:
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

V skripte využite cyklus `for` a/alebo `while`.

2.9 Polia

Tak ako v iných programovacích jazykoch aj v PHP je každý prvok poľa identifikovaný svojím indexom, resp. kľúčom (budeme používať oba termíny). Kľúč uvádzame v hranatých zátvorkách, napr. `$pole[7]`. Tento kľúč môže byť celé číslo alebo reťazec.

Číselné indexy

Číselné indexy štandardne začínajú od 0. Takéto pole môžeme naplniť postupným pridávaním prvkov do poľa v tvare `$pole[] = hodnota` alebo zadaním celého poľa v tvare `$pole = [hodnota1, hodnota2, ..., hodnotaN]`.



PRÍKLAD 2.31

Naplňte pole `$mena` postupne prvkami *Adam*, *Barbora*, *Cyril* a *Dano*. Pole `$tyzden` naplňte zadaním celého poľa. V poli budú dni od *pondelka* do *piatka*.

```

$mena[] = 'Adam';
$mena[] = 'Barbora';
$mena[] = 'Cyril';
$mena[] = 'Dano';
$tyzden = ['pondelok', 'utorok', 'streda', 'stvrtok', 'piatok'];

```

02/polia31.p
hp

ÚLOHA 2.32

Doplňte pole `$tyzden` o dni *sobota* a *nedela*.

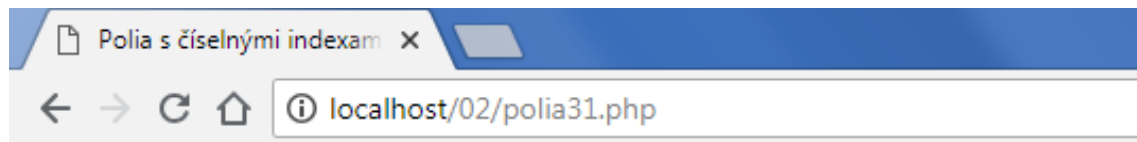
Obsah poľa môžeme vypísať pomocou príkazu `print_r(pole)`, kde `pole` môže byť premenná typu pole, alebo aj priamo zadané pole napr. `['a', 'b', 'c', 'd']`.

PRÍKLAD 2.33

Vypíšme obsah polí `$mena` a `$tyzden` (z príkladu 2.29) pomocou funkcie `print_r()`.

```
print_r($mena);  
echo "<br>\n";  
print_r($tyzden);
```

V prehliadači sa nám zobrazí:



```
Array ( [0] => Adam [1] => Barbora [2] => Cyril [3] => Dano )  
Array ( [0] => pondelok [1] => utorok [2] => streda [3] => štvrtok [4] => piatok )
```

A v zdrojovom kóde:

```
8 <body>  
9 Array  
10 (  
11     [0] => Adam  
12     [1] => Barbora  
13     [2] => Cyril  
14     [3] => Dano  
15 )  
16 <br>  
17 Array  
18 (  
19     [0] => pondelok  
20     [1] => utorok  
21     [2] => streda  
22     [3] => štvrtok  
23     [4] => piatok  
24 )  
25 </body>
```

Pomocou príkazu `echo` nevieme vypísať celé pole, teda nefunguje `echo $mena;`, ale vieme vypísať jednotlivé prvky poľa.



02/polia33.php

Príkaz `print_r()` budeme využívať len na pomocné výpisy.



02/polia34.p
hp

PRÍKLAD 2.34

Vypíšme 2. prvok poľa `$mena` (s kľúčom 1) a potom do ďalšieho riadku prvky s kľúčmi 0 a 3 oddelené medzerou. Ďalej zmeňme meno *Dano* na *Dana* (kľúč 3) a tento prvok znovu vypíšme.

zdrojový kód	výstup v prehliadači
<pre>echo \$mena[1]; echo "
\n"; echo "\$mena[0] \$mena[3]
\n"; \$mena[3] = 'Dana'; echo \$mena[3];</pre>	

Prechádzanie poľa

Najuniverzálnejší spôsob prechádzania poľa je pomocou funkcie `foreach()`.

Cyklus `foreach()` môžeme použiť dvomi spôsobmi. V prípade, že nás z poľa zaujímajú len hodnoty, použijeme

```
foreach ( pole as $hodnota ) { príkazy }.
```

Do premennej `$hodnota` sa postupne (v každom prechode cyklu) uložia jednotlivé **hodnoty poľa**. V prípade, že potrebujeme nejakým spôsobom pracovať aj s kľúčmi poľa, musíme použiť

```
foreach ( pole as $kluc => $hodnota ) { príkazy }.
```

Do premenných `$kluc` a `$hodnota` sa postupne (v každom prechode cyklu) uložia **kľúče a príslušné hodnoty poľa**.

Pri výpise poľa pomocou cyklu `foreach()` si už môžeme sami definovať, ako bude výpis vyzerať.



02/polia35.p
hp

PRÍKLAD 2.35

Vypíšme obsah poľa `$mena` tak, aby bolo každé meno na samostatnom riadku v prehliadači aj v zdrojovom kóde. Ďalej vypíšme obsah poľa `$tyzden` tak, aby sa v samostatných riadkoch vypísali jednotlivé dni v tvare kľúč = deň.

```
foreach($mena as $hodnota) {
    echo "$hodnota <br>\n";
}
foreach($tyzden as $kluc => $hodnota) {
    echo "$kluc = $hodnota <br>\n";
}
```

ÚLOHA 2.36



Vytvorte pole `$trieda` a naplňte ho krstnými menami študentov vašej triedy. Potom toto pole vypíšte tak, aby boli všetky mená v jednom riadku a za každým bola čiarka a medzera (aj za posledným menom).

Reťazcové kľúče - Asociatívne polia

Okrem číselných kľúčov môžu mať polia aj reťazcové kľúče. Takéto polia nazývame **asociatívne**. Takéto pole naplníme postupným pridávaním prvkov do poľa v tvare `$pole['kľúč'] = hodnota` alebo zadaním celého poľa v tvare `$pole = ['kľúč1' => hodnota1, 'kľúč2' => hodnota2, 'kľúč3' => hodnota3, ...]`.

PRÍKLAD 2.37



Vytvoríme pole `$osoba` s kľúčmi `meno`, `priezvisko`, `rok narodenia`, `mobil`, `mail` a hodnotami `Jožko`, `Mrkvička`, `2000`, `0999123456`, `jozko@mrkvicka.sk`. Potom vypíšme prvok s kľúčom `priezvisko`.

02/polia37.php

```
$osoba['meno'] = 'Jožko';
$osoba['priezvisko'] = 'Mrkvička';
$osoba['rok narodenia'] = 2000;
$osoba['mobil'] = '0999123456';
$osoba['mail'] = 'jozko@mrkvicka.sk';

echo $osoba['priezvisko'];
```

PRÍKLAD 2.38



Vypíšme vetu "Vitaj meno priezvisko.", kde `meno` je prvok poľa `$osoba` s kľúčom `meno` a `priezvisko` je prvok s kľúčom `priezvisko`.

02/polia38.php

Keď príklad vyriešime podobne ako v príklade 2.31, teda zadáme

```
echo "Vitaj $osoba['meno'] $osoba['priezvisko'].<br>\n";
```

tak nám prehliadač vypíše chybu, lebo prvok poľa nemôžeme priamo vypísať v reťazci, ak je kľúčom reťazec. Teda funguje

```
echo "Prvé meno je $mena[0].";
```

ale nefunguje

```
echo "Volám sa $osoba['meno'].";
```

Môžeme to vyriešiť rozdelením reťazca


```
echo "Volám sa " . $osoba['meno'] . " .";
```

alebo využijeme zložené zátvorky `{ }`, ktoré nám vyhodnotia v nich uzatvorený výraz

```
echo "Volám sa {$osoba['meno']}. .";
```

My budeme využívať druhý spôsob (zložené zátvorky), lebo pri zrefazovaní sa často zabúda na písanie bodiek a tým vznikajú chyby.

Správne riešenie nášho príkladu je teda:

```
echo "Vitaj {$osoba['meno']} {$osoba['priezvisko']}.<br>\n";
```



ÚLOHA 2.39

Vytvorte pole `$tovar` zadaním celého poľa s kľúčmi `Made Zigy`, `Sphero BB-8 Star Wars`, `LEGO Boost 17101`, `OZOBOT 2.0 BIT` a hodnotami `38.90`, `100.90`, `141.90`, `65.90`. Potom vypíšte všetky prvky poľa, každé v novom riadku, v tvare "kľúč - hodnota €".

2.10 Ladenie a analyzovanie chýb

Každý programátor robí chyby. Je však dôležité naučiť sa čítať chybové správy a potom chyby opravovať. Niekedy aj veľmi jednoduchá chyba spôsobí problém (chybu), no chybová správa PHP interpretera nemusí byť jasná, resp. sa z nej nedá dobre určiť, o akú chybu ide.

Jednou z pomôcok pri hľadaní chýb je aj vizuálne zobrazovanie jednotlivých častí kódu v editore (kľúčové slová, reťazce, čísla, premenné atď.). Ďalšou vizuálnou pomôckou je zobrazovanie párov zátvoriek. Keď sa k niektorej nastavíme, tak sa spolu s koncovou (začiatočnou) zvýraznia. Takto môžeme skontrolovať, či máme správne uzátvorkovaný kód. Ak sa nič nezvýrazní, tak zrejme k danej zátvorke chýba dvojica.

Aké sú časté chyby?

Chýbajúca ;



PRÍKLAD 2.40

Spustíte v prehliadači súbor `02/chyby01.php`. Čo sa zobrazí?

02/chyby01.p
hp

Zrejme sa vypíše nasledujúca správa:

Parse error: syntax error, unexpected 'číslo' (T_VARIABLE), expecting ';' or ';' in D:\...\02\chyby01.php on line 12

Táto správa hovorí, že na 12. riadku súboru je chyba - že sa neočakávalo '\$cislo' (premenná), ale očakáva sa čiarka ',' alebo bodkočiarka ';'. A naozaj nám na 10. riadku chýba bodkočiarka a preto na 12. riadku neočakával premennú.

```
9  <?php
10 echo "<h2>Výpis reťazca s premennou</h2>"
11
12 $cislo = 123;
13 echo "Výsledok je $cislo";
14 ?>
```

ÚLOHA 2.41



Opravte chybu v súbore 02/chyby01.php a opravený kód si pozrite v prehliadači. Potom zmažte ; na 12 riadku a pozrite sa, aká chyba sa vypíše v prehliadači.

Opäť v kóde chýba ;. Vo výpise správy

Parse error: syntax error, unexpected 'echo' (T_ECHO) in D:\...\02\chyby01.php on line 13

sa dozvieme, že sa neočakával príkaz echo, avšak už chýba časť expecting, čiže nevieme, čo sa namiesto neho očakávalo. To znamená, že pri rovnakej chybe nemusíme dostať rovnakú chybovú správu. V prípadoch, keď sa v chybovej správe uvádza, čo sa neočakávalo, treba hľadať zdroj chyby niekde **pred uvedeným neočakávaným textom**.

Chýbajúce ukončenie alebo začatie reťazca

PRÍKLAD 2.42



Spustíme v prehliadači súbor 02/chyby02.php. Čo sa zobrazí?

02/chyby02.php

Na 10. riadku nemáme ukončený reťazec, čo nám vizuálne zobrazí aj editor:

```
9  <?php
10 echo "<h2>Výpis reťazcov</h2>;
11
12 echo "Vypisujem reťazec v úvodzovkách.<br>";
13 echo 'Vypisujem reťazec v apostrofoch.<br>';
14
15 ?>
```

Vidíme, že príkaz echo na 12. riadku je sivý, hoci príkazy editor píše modrou farbou a reťazec, ktorý je na 12. riadku nie je sivý, ako zvyknú byť reťazce, ale čierny. Chybová správa, ktorá sa zobrazí, však nič nehovorí o chýbajúcej úvodzovke ("):

Parse error: syntax error, unexpected 'Vypisujem' (T_STRING), expecting ',' or ';' in D:\...\02\chyby02.php on line 12

Už vieme, že chybu máme hľadať pred riadkom 12.



ÚLOHA 2.43

Opravte chybu v súbore `chyby02.php` a opravený kód si pozrite v prehliadači. Zmažte iné úvodzovky (") alebo apostrofy (') (skúšajte ich zmazať na začiatku reťazca aj na konci reťazca) a pozorujte, aké chyby sa zobrazujú.

Ak zmažeme koncovú úvodzovku (") reťazca na 12. riadku, zobrazí sa chyba:

Parse error: syntax error, unexpected end of file, expecting variable (T_VARIABLE) or \${ (T_DOLLAR_OPEN_CURLY_BRACES) or { (T_CURLY_OPEN) in D:\...\02\chyby02.php on line 18

Táto chyba hovorí, že sa neočakával koniec súboru. Ak sa pozrieme na kód v editore, tak vidíme, že až po koniec je sivý, dokonca aj HTML značky. V takom prípade musíme hľadať, čo sme neukončili. Nemusí to byť len neukončený reťazec, ale aj neukončený cyklus, teda chýba koncová zátvorka, neukončená `if` alebo `else` vetva,...

```
9 <?php
10 echo "<h2>Výpis reťazcov</h2>";
11
12 echo "Vypisujem reťazec v úvodzovkách.<br>";
13 echo 'Vypisujem reťazec v apostrofoch.<br>';
14
15 ?>
16 </body>
17 </html>
18
```

Zámena porovnania s priradením

Zoberme si kód z príkladu 2.22. Ak by sme v podmienke zamenili porovnanie (`==`) za priradenie (`=`), PHP nevyhlási chybu, ale daný príkaz vykoná a jeho úspešné vykonanie nahradí logickou hodnotou `TRUE`.

```
$den = 'streda';
if ($den = 'sobota' || $den == 'nedela') {
    echo 'Krásny víkend!';
} else {
    echo 'Šup do školy';
}
```

Ak by sme porovnania v podmienke vymenili (`if ('sobota' = $den || 'nedela == $den') { }`), PHP by vyhlásilo chybu.

Ďalšie časté chyby:

- chýbajúca zátvorka (väčšinou koncová) `) }`
- chýbajúci znak `$` pri premennej
- chýbajúci operátor zretáženia `.`

ÚLOHA 2.44



Otvorte súbor `chyby.php` (z priečinku 02). Nájdite v ňom chyby a opravte ich. Sledujte, čo sa vám vypisuje na stránke.

02/chyby.php

2.11 Metodické pokyny pre učiteľa



CIEĽ

Cieľom tejto kapitoly je oboznámenie sa s väčšinou základných štruktúr jazyka PHP. Predpokladáme, že s väčšinou pojmov sa už študenti stretli pri inom programovacom jazyku. Učiteľ to môže využiť a poukázať na podobnosti alebo odlišnosti.



VÝKLAD

Druhá časť je zameraná na:

- vypisovanie údajov pomocou PHP,
- premenné a konštanty,
- prácu s reťazcami,
- operátory,
- vetvenie,
- cykly,
- polia,
- a ladenie chýb.

Výklad sa strieda s riešenými príkladmi a samostatnými úlohami pre študentov.

[Príkaz echo, reťazce \(1\)](#)

K úlohe 2.4: Očakávame, že študenti použijú element `
` alebo `<p>`.

[Komentáre](#)

Poznámka: Treba si dať pozor na to, aby sme nepoužívali HTML komentáre v PHP kóde. Ak chceme do výslednej (vygenerovanej) stránky vložiť HTML komentár, musíme použiť reťazce (ako pri iných HTML elementoch).

[Reťazce \(2\)](#)

Echo môžeme zapísať aj ako funkciu s jedným parametrom, teda `echo (čosi)`, my však tento spôsob nebudeme používať.

[Logické operátory](#)

Prehľadnú tabuľku o prioritách nájdete na

<http://php.net/manual/en/language.operators.precedence.php>.

[Porovnávacie operátory](#)

`echo 3 > 5;` - nevypíše sa nič, lebo `FALSE` sa prekonvertuje na prázdny reťazec

Príkaz if

```
if (podmienka) {  
    // príkazy  
}
```

Ak sa vykonáva len jeden príkaz, môžeme zložené zátvorky vynechať. My ich však budeme písať vždy, aby to študentov nemýlilo a nezabúdali písať zátvorky. Vyhneme sa tak prípadným chybám. Napríklad ak by sme chceli pridať ďalší príkaz do niektorej časti (IF, ELSE) a zabudli pridať zložené zátvorky.

Zámer pridať ďalší príkaz do časti ELSE - príkaz sa vypíše vždy.	Zámer pridať ďalší príkaz do časti IF - príkaz spôsobí chybu "Syntax error".
<pre>\$body = 85; if (\$body > 60) echo 'Test si spravil'; else echo 'Test si nespravil'; echo 'Môžeš to skúsiť znovu';</pre>	<pre>\$body = 85; if (\$body > 60) echo 'Test si spravil'; echo "Získal si \$body bodov"; else echo 'Test si nespravil';</pre>

Cykly

```
for (inicializácia; podmienka; príkaz) {  
    // príkazy  
}
```

Poznámka: Pri cykle for je najväčšia zmena oproti jazyku Pascal. Treba si dať pozor hlavne na časť **príkaz** – tú v Pascale nemusíme riešiť, pretože Pascal automaticky zvyšuje premennú cyklu o jedna. Ak sa študenti stretli s for cyklom v jazyku Python (a v iných jazykoch zatiaľ neprogramovali), je pre nich zápis cyklu for v PHP neznámy. Je vhodné venovať mu viac pozornosti a vysvetliť študentom význam všetkých troch zložiek v zátvorke (**inicializácia, podmienka, príkaz**). Taktiež je potrebné upozorniť ich na existenciu a význam zložených zátvoriek { }, ktoré z Pythonu nepoznajú a nie sú zvyknutí označovať začiatok a koniec bloku **príkazy**.

Ak chceme v cykle vykonať viac ako jeden príkaz, musíme ich dať do blokov pomocou zložených zátvoriek { }, podobne ako pri príkaze if. Ak však potrebujeme vykonať len jeden príkaz, zložené zátvorky môžeme vynechať. Podobne ako pri if.

Polia

Rôzne kľúče (číselné a reťazcové) môžeme vzájomne kombinovať v jednom poli. Zámerne to však neuvádzame. Číselné indexy zvyčajne začínajú nulou (ak pridávame prvky do poľa bez udania indexu). V prípade, že nevedieme index, hodnota je automaticky priradený index maximum+1. Teda PHP nevyhľadáva prípadné medzery medzi indexami, ale novému prvku priradí maximum+1. V prípade, že by sme mali definované indexy poľa 0, 1, 3 a vkladáme nový prvok bez udania indexu, potom mu PHP pridelí index 4.

Pole vieme indexovať od ľubovoľného čísla, ale zámerne to v materiáli neuvádzame, lebo s číselnými indexmi poľa nebudeme veľmi pracovať.

Okrem funkcie `foreach()` môžeme pole prechádzať už známymi cyklami `for` prípadne `while`. Tie však môžeme použiť len vtedy, ak sú indexy poľa číselné a spojité. Pri týchto cykloch totiž zvyčajne použijeme funkciu `count()`, ktorá vráti počet prvkov poľa. Týmto zápisom sa však nebudeme venovať, nakoľko budeme väčšinou pracovať s reťazcovými kľúčmi.

Využívanie zložených zátvoriek v echo

Do zložených zátvoriek môžeme uzavrieť jednoduchú premennú, prvok poľa alebo vlastnosť objektu (ak využívame objektové programovanie). Znak `$` musí nasledovať hneď po `{`, inak výpis nebude korektný. Prvok poľa s reťazcovým kľúčom sa korektne vypíše len keď je uzavretý v zložených zátvorkách, teda: `echo "Volám sa {$osoba['meno']}. ";`

Zložené zátvorky v príkaze `echo` môžeme napr. využiť, keď chceme odlišiť názov premennej od okolitého textu, napríklad, ak by sme chceli vygenerovať vetu z nasledujúcich premenných

```
$opakuj = 5;  
$aky = 'dvoj';  
echo "Urob $opakuj-krát $akyskok.<br>\n";
```

tak premennú `$opakuj` rozpozná, ale premennú `$aky` nie (dostaneme upozornenie, že premenná `akyskok` nie je definovaná). Preto ju musíme uzavrieť do zložených zátvoriek:

```
$opakuj = 5;  
$aky = 'dvoj';  
echo "Urob $opakuj-krát {$aky}skok.<br>\n";
```

Ladenie a analyzovanie chýb

K príkladu 2.42: Tým, že sme reťazec na 10. riadku neuzavreli, pokračoval vlastne ďalej až po najbližšiu `"`. A za ňou PHP interpretér našiel slovo *Vypisujem*, ktorému nerozumie, preto v chybovej správe píše, že ho neočakáva.

3 PHP - SPRACOVANIE FORMULÁROV (1)

V tejto časti si ukážeme, ako pomocou formulárov môžeme získať údaje od používateľov stránok a ako s týmito údajmi ďalej pracovať.

3.1 Odoslanie formulára, tlačidlo

Vytvoríme nasledujúci jednoduchý formulár s tlačidlom na odoslanie formulára. Formulár sa bude odosielať metódou `POST`.

```
<form method="post">
  <input type="submit" name="tlacidlo" value="Odošli údaje">
</form>
```

Pripomeňme si, že aby sme mohli odoslať formulár, musí v ňom byť tlačidlo typu submit

03/formular01.php

ZAPAMÄTAJTE SI

Metóda POST:

- údaje z formulára sú posielané cez HTTP odpoveď na požiadavku v asociatívnom poli `$_POST` (narozdiel od metódy GET, kde sa údaje získavajú z URL adresy)
- môžu sa posielajú aj citlivé údaje, lebo nie sú viditeľné
- dĺžka (veľkosť) údajov je obmedzená len nastaveniami webového servera

Čo sa stane po odoslaní formulára? Údaje z formulára sa pošlú súboru, ktorý je špecifikovaný v atribúte `action` pre `<form>`. Ak tento atribút nie je uvedený, tak sa pošlú tomu istému súboru, teda súboru, v ktorom je formulár. My budeme využívať práve takéto spracovanie formulárov.

ÚLOHA 3.1

Zobrazme výslednú stránku v prehliadači a otestujme. Čo sa zmení na stránke po odoslaní formulára (kliknutí na tlačidlo)?

Po odoslaní formulára metódou `POST` sa naplní asociatívne pole `$_POST` všetkými zadanými hodnotami z formulára (vrátane hodnoty odosiacieho tlačidla). Pole `$_POST` má tvar [kľúč1 => hodnota1, kľúč2 => hodnota2, kľúč3 => hodnota3, ...], kde kľúčmi sú názvy prvkov formulára (atribúty `name`) a hodnotami sú odoslané údaje z príslušných prvkov formulára.

```
<input type="submit" name="tlacidlo"
value="Odošli údaje">
```

=>

```
$_POST['tlacidlo']
```

PRÍKLAD 3.2

Vypíšme hodnotu odoslaného tlačidla. Všimajme si, čo sa zobrazí pred a po odoslaní formulára (kliknutím na tlačidlo). Výpis umiestnime podľa toho, či chceme mať výpis pred formulárom alebo za.

03/formular02a.php
03/formular02b.php

Ak stránku (súbor) s formulárom viackrát upravujeme a znovu zobrazujeme v prehliadači, odporúčame nepoužívať tlačidlo Obnoviť ale kliknúť do panela s adresou a potvrdiť klávesom Enter.

```
<?php
echo $_POST['tlacidlo'];
?>
<form method="post">
  <input type="submit" name="tlacidlo" value="Odošli údaje">
</form>
```

```
<form method="post">
  <input type="submit" name="tlacidlo" value="Odošli údaje">
</form>
<?php
echo $_POST['tlacidlo'];
?>
```

Pri prvom zobrazení stránky sa nám zobrazí upozornenie, že neexistuje kľúč `tlacidlo`. Tým sa myslí kľúč v poli, v našom prípade je to kľúč `tlacidlo` z poľa `$_POST`. Po odoslaní formulára sa upozornenie nezobrazí.

Spôsobené je to tým, že pri prvom zobrazení stránky je ešte pole `$_POST` prázdne, teda sa snažíme vypísať hodnotu neexistujúceho prvku poľa. Aby sme predišli zobrazeniu takejto "škaredej" správy, pred samotným výpisom premennej (prvku poľa) najprv otestujeme, či taká premenná existuje.

Na overenie existencie premennej, resp. prvku poľa, nám slúži funkcia `isset()`. Funkcia vráti hodnotu `TRUE` alebo `FALSE` podľa toho, či daná premenná (index poľa) existuje.



03/formular0
3.php

PRÍKLAD 3.3

Doplňme do predchádzajúceho príkladu kontrolu existencie premennej, a tým odstráňme zobrazenie upozornenia.

```
<?php
if (isset($_POST['tlacidlo'])) {
  echo $_POST['tlacidlo'];
  // echo 'Stlačil si tlačidlo';
}
?>
<form method="post">
  <input type="submit" name="tlacidlo" value="Odošli údaje">
</form>
```

3.2 Textové pole - INPUT a TEXTAREA



03/formular0
4.php

PRÍKLAD 3.4

Doplňme do formulára jednoduché textové pole, do ktorého bude používateľ zadávať údaje. Tie následne (po odoslaní formulára) vypíšeme.

```

<?php
if (isset($_POST['meno'])) {
    echo $_POST['meno'];
    // echo 'meno: ' . $_POST['meno'];
}
?>
<form method="post">
    <label for="meno">Meno:</label> <input type="text" name="meno"
id="meno">
    <input type="submit" name="tlacidlo" value="Odošli údaje">
</form>

```

Rovnako by sme pracovali aj s položkou s heslom (pole typu password).

V prípade, že používateľ nezadá žiadne meno a formulár odošle, nevypíše sa nič, ani chyba, resp. upozornenie. Dôvodom je to, že PHP interpret nastaví ako hodnotu textového poľa prázdny reťazec a ten priradí do `$_POST['meno']`. Teda funkcia `isset()` vráti hodnotu `TRUE`. Ak do príkazu `echo` pridáme nejaký reťazec, zistíme, že PHP danú časť kódu naozaj vykoná (napr. `echo 'meno: ' . $_POST['meno'];`).

PRÍKLAD 3.5

Upravme formulár tak, že ak používateľ meno nezadá, vypíše sa správa *Nezadal si meno!*



03/formular05.php

```

if (isset($_POST['meno']) && ($_POST['meno'] != '')) {
    echo 'meno: ' . $_POST['meno'];
} else {
    echo 'Nezadal si meno!';
}

```

Pri odoslaní formulára toto riešenie funguje, ale bohužiaľ, správa sa zobrazí aj pri prvom zobrazení stránky (pred odoslaním formulára). A to nie je dobre.

Problémom je, že naša kontrola mena sa vykoná vždy pri zobrazení stránky, teda aj keď nie je odoslaný formulár. Preto musíme náš kód doplniť o ďalšiu podmienku, ktorá by sa mala stať základnou pri spracovaní formulára - kontrola stlačenia odosielacieho tlačidla (teda existencie premennej `$_POST['tlacidlo']`) - `if (isset($_POST['tlacidlo'])) { }`. A všetky ďalšie kontroly týkajúce sa zadaných hodnôt do formulára by sa mali vykonať len keď je podmienka splnená.

PRÍKLAD 3.6

Vylepšime kontrolu formulára z príkladu 3.5.



03/formular06.php

```

if (isset($_POST['tlacidlo'])) {
    if (isset($_POST['meno']) && ($_POST['meno'] != '')) {
        echo 'meno: ' . $_POST['meno'];
    } else {
        echo 'Nezadal si meno!';
    }
}

```

Ak sme do textového poľa nezadali nič, program nás na to síce upozorní, ale len vtedy, ak odošleme formulár. Pri prvom zobrazení stránky sa zobrazí len formulár bez upozornenia.

V prípade, že vo formulári nepoužijeme jednoriadkové textové pole, ale viacriadkové textové pole pomocou elementu `<textarea>`, spracovanie vstupu bude rovnaké.



03/formular0
7.php

PRÍKLAD 3.7

V príklade 3.6 zmeňme položku meno na adresa a prvok input na textarea.

```
<?php
if (isset($_POST['tlacidlo'])) {
    if (isset($_POST['adresa']) && ($_POST['adresa'] != '')) {
        echo 'adresa: ' . $_POST['adresa'];
    } else {
        echo 'Nezadal si adresu!';
    }
}
?>

<form method="post">
    <label for="adresa">Adresa:</label>
    <textarea name="adresa" id="adresa"></textarea><br>
    <input type="submit" name="tlacidlo" value="Odošli údaje">
</form>
```

Vidíme, že vo funkčnosti spracovania formulára sme nič nezmenili, len sme vymenili texty, ktoré sa majú vypísať a zmenili sme názvy položiek.



ÚLOHA 3.8

Vytvorte formulár s tromi textovými poľami pre meno, priezvisko a mesto. Ak nie sú zadané všetky údaje, vypíše sa správa *Nezadal si všetky údaje!*, inak sa vypíše správa *Volám sa ... a bývam v ...* (ako na nasledujúcom obrázku)

Volám sa Janko Hraško a bývam v Haluškovo.
Meno:
Priezvisko:
Mesto:
Odošli údaje

ÚLOHA 3.9



Vytvorte formulár s dvoma textovými poľami pre dve strany obdĺžnika. Ak nie sú zadané všetky údaje, vypíše sa správa *Nezadal si obe strany obdĺžnika!*, inak sa vypíše správa *Obdĺžnik so stranami ... , ... má obvod ... a obsah ...*.

3.3 Výberová ponuka - SELECT

Pri výberovej ponuke (element `<select>`) zvyčajne vyberáme jednu hodnotu z viacerých. Hodnota celého elementu po odoslaní formulára sa bude rovnať hodnote vybranej položky - hodnote atribútu `value`.

PRÍKLAD 3.10



03/formular10.
php

Vložme do stránky výberovú ponuku a po odoslaní zobrazme jej hodnotu. Všímajme si hodnoty vo výberovej ponuke a hodnoty, ktoré sa naozaj vypíšu po spracovaní formulára.

```
<?php
if (isset($ POST['tlacidlo'])) {
    if (isset($_POST['vyber']) && ($_POST['vyber'] != '')) {
        echo 'vyber: ' . $_POST['vyber'];
    } else {
        echo 'Nič si nevybral!';
    }
}
?>

<form method="post">
    <label for="vyber">vyber:</label>
    <select name="vyber" id="vyber">
        <option value="">0</option>
        <option value="10">1</option>
        <option value="20">2</option>
        <option value="30">3</option>
        <option value="40">4</option>
        <option value="50">5</option>
    </select><br>
    <input type="submit" name="tlacidlo" value="Odošli údaje">
</form>
```

Zapamätajte si: Pri spracovávaní hodnoty (value) si musíme uvedomiť, že všetky hodnoty sú reťazce, aj keď sú v nich uložené čísla. Ak sú to však korektné čísla, PHP si ich vie prerobiť (pretypovať), teda môžeme s nimi robiť matematické operácie.

Vo formulári sme chceli, aby používateľ mohol zvolil jednu z hodnôt 1 až 5. Ak by sme mali len tieto hodnoty v elementoch `<option>`, bola by automaticky zobrazená hodnota 1 (prvá v poradí). Ak by bola po odoslaní formulára vo výbere 1 (hodnota 10), tak by sme nevedeli rozlíšiť, či používateľ naozaj vybral položku 1 alebo odoslal formulár bez toho, aby niečo zvolil. Preto je vhodné, aby vo výberovej ponuke bola "prázdna" hodnota. My sme do ponuky zaradili aj položku `<option value="">0</option>`.



ÚLOHA 3.11

Vytvorte formulár, v ktorom bude výberová ponuka rôznych druhov pizze. V ponuke bude názov pizze spolu s jej cenou a po spracovaní formulára sa vypíše len zvolený názov pizze, prípadne správa, že sa nič nezvolilo. Môžete využiť tieto informácie: *Margherita - 4,50 €*, *Cardinale - 5,50 €*, *Funghi - 5,50 €*, *Hawai - 7,00 €*. Výberová ponuka môže vyzeráť nasledovne:

vyber pizzu: -----
Odošli údaje
Margherita - 4,50 €
Cardinale - 5,50 €
Funghi - 5,50 €
Hawai - 7,00 €

3.4 Prepínač - RADIOBUTTON

Inou možnosťou, ako na stránke vyberať jednu hodnotu z viacerých, sú prepínače - radiobuttony. Z HTML vieme, že ak chceme vyberať z viacerých prepínačov, musia mať rovnaké hodnoty atribútu `name`.

Samotné spracovanie prepínačov je rovnaké ako pri výberovej ponuke - hodnota zvoleného prepínača bude hodnotou celej skupiny prepínačov.

Pri odoslaní formulára sa prepínače správajú trochu inak ako pri výberovej ponuke. Ak totiž neoznačíme žiaden radiobutton, v poli `$_POST` nevznikne žiadny kľúč prislúchajúci danej skupine prepínačov (ani s prázdny reťazcom). Ten vznikne jedine vtedy, ak označíme niektorý radiobutton. Vďaka tomu môžeme podmienku zjednodušiť na: `if (isset($_POST['vyber'])) { }`.



PRÍKLAD 3.12

03/formular1
2.php

Nahradíme výberovú ponuku z príkladu 3.10 prepínačmi.

```
<?php
if (isset($_POST['tlacidlo'])) {
    if (isset($_POST['vyber'])) {
        echo 'vyber: ' . $_POST['vyber'];
    } else {
        echo 'Nič si nevybral!';
    }
}
?>

<form method="post">
    <label for="vyber1">1</label> <input type="radio" name="vyber"
value="10" id="vyber1"><br>
    <label for="vyber2">2</label> <input type="radio" name="vyber"
value="20" id="vyber2"><br>
```

```

<label for="vyber3">3</label> <input type="radio" name="vyber"
value="30" id="vyber3"><br>
<label for="vyber4">4</label> <input type="radio" name="vyber"
value="40" id="vyber4"><br>
<label for="vyber5">5</label> <input type="radio" name="vyber"
value="50" id="vyber5"><br>
<input type="submit" name="tlacidlo" value="Odošli údaje">
</form>

```

Vidíme, že spracovanie formulára je takmer rovnaké ako pri výberovej ponuke, len sme zjednodušili podmienku.

ÚLOHA 3.13

Upravte riešenie úlohy 3.11 tak, aby ste namiesto výberovej ponuky využili prepínače. Formulár môže vyzerať nasledovne:

```

vybral si pizzu: Cardinale
 Margherita - 4,50 €
 Cardinale - 5,50 €
 Funghi - 5,50 €
 Hawai - 7,00 €


```



3.5 Začiarkávacie políčko - CHECKBOX

Aby mohol používateľ označiť viacero hodnôt súčasne, používame začiarkávacie políčka (checkboxy). Ich spracovanie je v podstate rovnaké ako spracovanie prepínačov, ale každé políčko musíme kontrolovať samostatne. Každé začiarkávacie políčko má inú hodnotu atribútu `name`. Hodnoty atribútov `value` môžu byť rovnaké.

Začiarkávacie políčka sa správajú rovnako ako prepínače - ak ich neoznačíme, po odoslaní formulára nevznikne v poli `$_POST` príslušný kľúč.

Začiarkávacie políčka zvyčajne používame vtedy, ak používateľ nemusí zvoliť žiadnu hodnotu, alebo ich môže označiť viacero. Ak by sme chceli kontrolovať, či označil aspoň jednu (prípadne aspoň niekoľko), museli by sme si to spočítať, a teda kontrolovať všetky políčka, či boli/neboli označené.

PRÍKLAD 3.14

Nahradíme prepínače z príkladu 3.12 začiarkávacími políčkami. Využijeme aj predchádzajúci PHP kód, len ho viackrát skopírujeme, teda pre každé políčko vypíšeme, či bolo/nebolo vybraté.



03/formular14.php

```

<?php
if (isset($_POST['tlacidlo'])) {
    if (isset($_POST['vyber1'])) {

```

```

    echo "vyber 1: {$_POST['vyber1']} <br>";
} else {
    echo 'Nezadal si vyber 1!<br>';
}
if (isset($_POST['vyber2'])) {
    echo "vyber 2: {$_POST['vyber2']} <br>";
} else {
    echo 'Nezadal si vyber 2!<br>';
}
if (isset($_POST['vyber3'])) {
    echo "vyber 3: {$_POST['vyber3']} <br>";
} else {
    echo 'Nezadal si vyber 3!<br>';
}
if (isset($_POST['vyber4'])) {
    echo "vyber 4: {$_POST['vyber4']} <br>";
} else {
    echo 'Nezadal si vyber 4!<br>';
}
if (isset($_POST['vyber5'])) {
    echo "vyber 5: {$_POST['vyber5']} <br>";
} else {
    echo 'Nezadal si vyber 5!<br>';
}
}
?>

<form method="post">
    <label for="vyber1">1</label> <input type="checkbox" name="vyber1"
value="10" id="vyber1"><br>
    <label for="vyber2">2</label> <input type="checkbox" name="vyber2"
value="20" id="vyber2"><br>
    <label for="vyber3">3</label> <input type="checkbox" name="vyber3"
value="30" id="vyber3"><br>
    <label for="vyber4">4</label> <input type="checkbox" name="vyber4"
value="40" id="vyber4"><br>
    <label for="vyber5">5</label> <input type="checkbox" name="vyber5"
value="50" id="vyber5"><br>
    <input type="submit" name="tlacidlo" value="Odošli údaje">
</form>

```



PRÍKLAD 3.15

03/formular1
5.php

Zjednodušíme príklad 3.14 tak, že budeme vypisovať len tie začiarkávacie políčka, ktoré používateľ označil. Teda vynecháme `else` vetvy o nezadaní príslušného políčka.

```

if (isset($_POST['tlacidlo'])) {
    if (isset($_POST['vyber1'])) {
        echo "vyber 1: {$_POST['vyber1']} <br>";
    }
    if (isset($_POST['vyber2'])) {
        echo "vyber 2: {$_POST['vyber2']} <br>";
    }
    if (isset($_POST['vyber3'])) {
        echo "vyber 3: {$_POST['vyber3']} <br>";
    }
}

```

```
if (isset($_POST['vyber4'])) {
    echo "vyber 4: {"$_POST['vyber4']} <br>";
}
if (isset($_POST['vyber5'])) {
    echo "vyber 5: {"$_POST['vyber5']} <br>";
}
}
```

ÚLOHA 3.16

Vytvorte formulár, pomocou ktorého sa bude dať zisťovať, v akých programovacích jazykoch sa používatelia učili programovať. Napr. Pascal, Python, C++, C#, Java, PHP, Imagine, Scratch,... Po spracovaní formulára sa vypíšu len tie jazyky, ktoré používateľ zvolil.



ÚLOHA 3.17

Vytvorte formulár, pomocou ktorého bude možné nakombinovať si vlastnú pizzu. Budú sa môcť zvoliť prísady, a tiež sa bude dať zvoliť, či to má byť malá alebo veľká pizza. Po odoslaní formulára sa vypíšu zvolené údaje. Formulár po odoslaní môže vyzeráť nasledovne:



vybral si si tieto prísady: vajičko, ananás, kukurica, kuracie mäso,
vybral si pizzu: veľká

šunka
saláma
slanina
vajičko
šampiňóny
cibuľa
olivy
ananás
kukurica
tuniak
kuracie mäso
krevety
paradajky

malá | veľká

Odošli údaje

3.6 Metodické pokyny pre učiteľa



CIEĽ

Cieľom tejto kapitoly je oboznámenie so spracovaním najpoužívanejších formulárových elementov.



VÝKLAD

Táto časť je zameraná na:

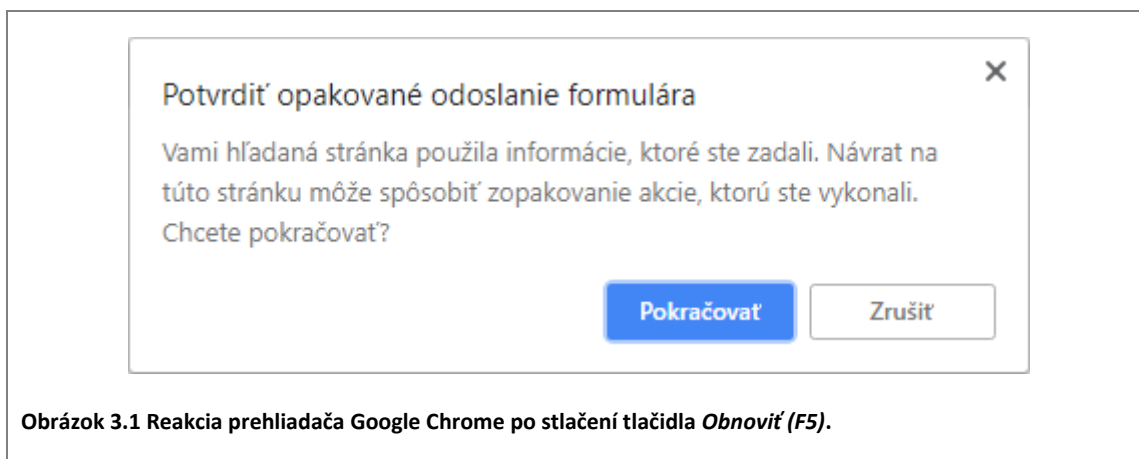
- odosielanie formulára,
- a na spracovanie väčšiny formulárových elementov.

Výklad sa strieda s riešenými príkladmi a samostatnými úlohami pre študentov.

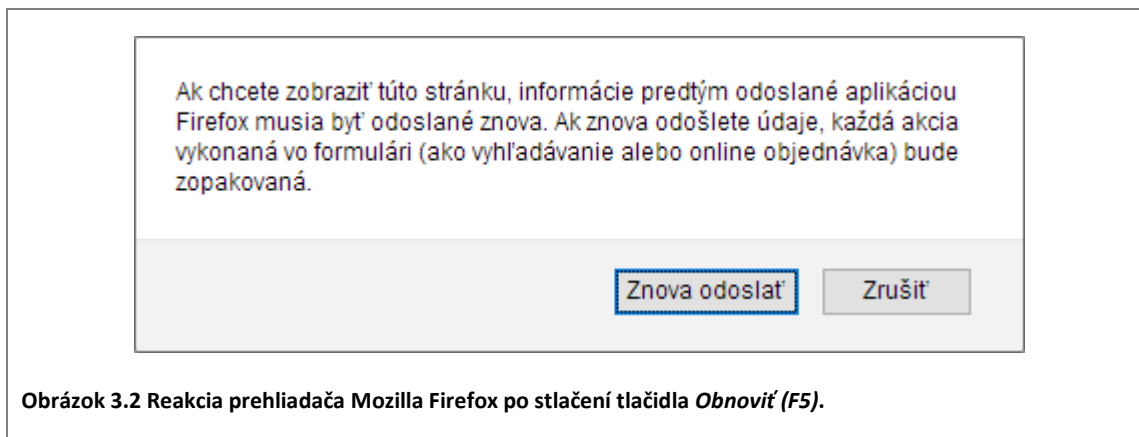
Študentov treba upozorniť, že formulár musí vždy obsahovať tlačidlo typu submit, aby sa dal odoslať.

K načítaniu stránky, resp. obnoveniu stránky s formulárom:

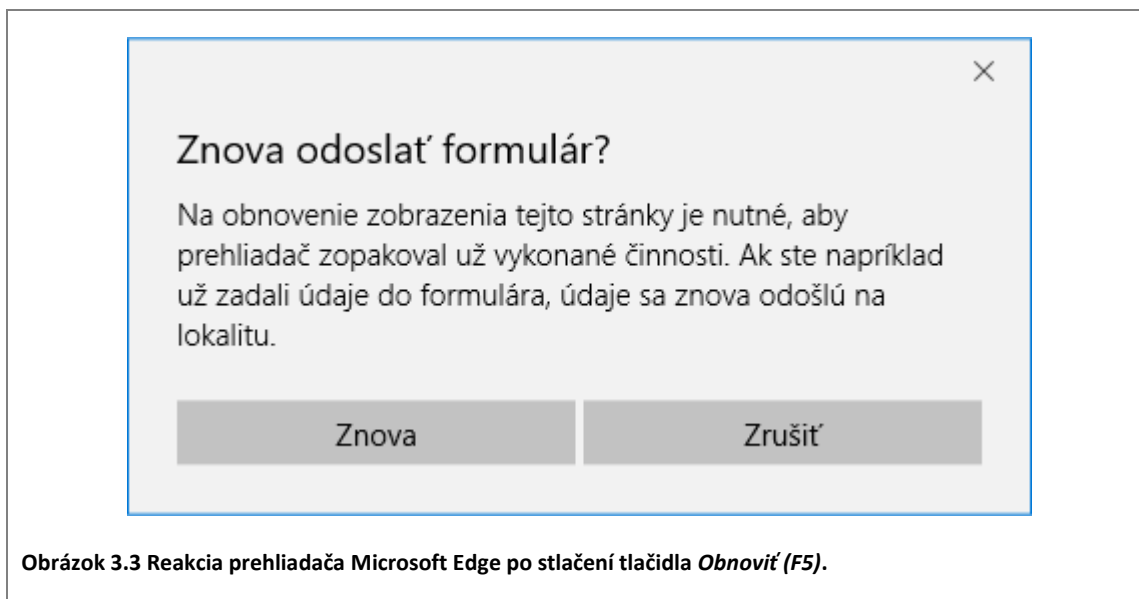
Treba upozorniť študentov, že načítať tú istú stránku s formulárom, keď klikneme na tlačidlo *Obnoviť (F5)*, je iné ako keď klikneme do panela s adresou a potvrdíme klávesom *Enter*. A to hlavne vtedy, keď sme formulár odoslali. Prehliadač si totiž pamätá, či stránka dostala nejaké vstupy (z odoslaného formulára). Ak áno, tak pri obnovení stránky cez tlačidlo *Obnoviť (F5)* sa prehliadač spýta, či sa majú znovu odoslať údaje z formulára.



Obrázok 3.1 Reakcia prehliadača Google Chrome po stlačení tlačidla *Obnoviť (F5)*.



Obrázok 3.2 Reakcia prehliadača Mozilla Firefox po stlačení tlačidla *Obnoviť* (F5).



Obrázok 3.3 Reakcia prehliadača Microsoft Edge po stlačení tlačidla *Obnoviť* (F5).

Preto, ak chceme znovu načítať stránku v prehliadači, odporúčame **nepoužívať** tlačidlo *Obnoviť* (F5), ale kliknúť do panela s adresou a potvrdiť klávesom *Enter*.

K spracovaniu formulára:

PHP kód, ktorý spracováva formulár môže byť umiestnený v inom súbore ako HTML formulár, alebo môže byť v rovnakom súbore ako HTML formulár. V učebnici budeme využívať len druhý prístup (formulár a PHP kód v jednom súbore). Tento prístup nám umožňuje lepšie reagovať na zadané údaje a podľa potreby zobrazovať/skrývať formulár, prípadne zobrazovať pri formulári chybové správy. Pri prvom prístupe (HTML formulár a PHP kód v samostatných súboroch) by sa napr. chybové správy pri formulári zobrazovali komplikovanejším spôsobom. Museli by sme si ich nejako posilať alebo ukladať - a to je niečo, čo sa ešte v učebnici neukazovalo.

`$_POST` je superglobálna premenná, ktorá je vždy prístupná (aj keď je prázdna). Môžeme k nej hocikedy pristupovať, a to z ľubovoľnej vlastnej funkcie, triedy či súboru.

K časti 3.1:

Pre tieto ukážky je jedno, či použijeme tlačidlo cez element `<input>` alebo cez element `<button>`.

K úlohe 3.1:

Treba upozorniť študentov, že sa možno zdá, že sa nič nedeje. V skutočnosti sa odošle formulár tomu istému súboru a ten sa znovu načíta. Treba sledovať ikonu obnovenia stránky.

K elementu SELECT:

Ak atribút value neexistuje, hodnotou bude text, ktorý sa pri danej položke zobrazuje, teda ak sa hodnoty a texty zhodujú, nemuseli by sa atribúty `value` uvádzať.

4 PHP - ZÁKLADNÁ SYNTAX A ŠTRUKTÚRY (2)

V tejto časti doplníme základné informácie o jazyku PHP. Ukážeme si prácu s náhodnými číslami, dátumom a časom, naučíme sa vytvárať vlastné funkcie a využívať viaceré PHP vsuvky.

4.1 Náhodné čísla

Ak budeme potrebovať vygenerovať náhodné číslo, môžeme použiť funkciu `rand()`. Funkcia generuje náhodné celé číslo z intervalu $<0, \text{getrandmax()}>$. Ak potrebujeme upraviť rozsah, použijeme `rand(min, max)`. Vtedy funkcia vráti náhodné **celé číslo** z intervalu $<\text{min}, \text{max}>$.

PRÍKLAD 4.1

Vypíšme 5 náhodných čísel pod seba. Ďalej vypíšme náhodné číslo z intervalu $<5, 20>$. Nakoniec vypíšme náhodné číslo z intervalu $<10, 100>$, pričom to môže byť len násobok čísla 5.



04/nahodne.ph
P

```
<?php
echo "<h1>Náhodné čísla</h1>\n";
for ($i = 1; $i <= 5; $i++) {
    echo rand() . "<br>\n";
}
echo "<h1>Náhodné číslo z intervalu <5, 20></h1>\n";
echo rand(5, 20) . "<br>\n";
echo "<h1>Náhodné číslo z intervalu <10, 100>, ale len násobky
5</h1>\n";
$a = rand(1, 19) * 5 + 5;
echo "$a<br>\n";
?>
```

POZNÁMKA

Okrem funkcie `rand()` môžeme na generovanie náhodných čísel použiť aj funkciu `mt_rand()`. Táto funkcia generuje čísla pomocou iného algoritmu, vracia lepšie výsledky a je 4x rýchlejšia ako `rand()`. V prípade, že chceme použiť `mt_rand(min, max)`, musíme sa uistiť, že `min <= max`, inak sa vypíše upozornenie.



4.2 Dátum a čas

Na webových stránkach sa často vyskytujú dátumové informácie. Napr. informácia o aktuálnom dátume, informácia o poslednom prihlásení, o tom, kedy bude dostupný tovar, v ktorých dňoch a časoch sa premieta daný film atď. Tieto informácie môžu byť zapísané rôznymi spôsobmi.



ÚLOHA 4.2

Prezrite si niekoľko webových stránok a nájdite na nich dátumové, prípadne časové údaje. Akým spôsobom (v akom formáte) boli zapísané?

Aby sme zistili, či je daný dátum platný, použijeme funkciu `checkdate(mesiac, den, rok)`. Funkcia vracia `TRUE` alebo `FALSE`.

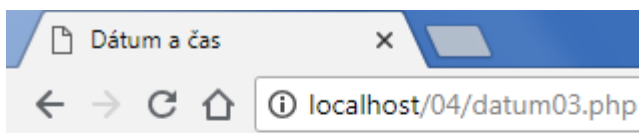


PRÍKLAD 4.3

Overme platnosť dátumov 6.8.2018 a 31.6.2018.

```
echo "dátum 6.8.2018 ";
if (checkdate(8,6,2018)) {
    echo "je platný<br>\n";
} else {
    echo "nie je platný<br>\n";
}
echo "dátum 31.6.2018 ";
if (checkdate(6,31,2018)) {
    echo "je platný<br>\n";
} else {
    echo "nie je platný<br>\n";
}
```

Tento kód vypíše:



Overenie dátumu

dátum 6.8.2018 je platný
dátum 31.6.2018 nie je platný



ÚLOHA 4.4

Vygenerujte 10 náhodných dátumov (pre deň náhodné číslo od 1 do 31, pre mesiac náhodné číslo od 1 do 12 a pre rok náhodné číslo od 1800 po 2100) a overte a vypíšte ich platnosť.

Na formátovanie dátumu a/alebo času použijeme funkciu `date(format)`. Parameter `format` je povinný a definuje, v akom formáte funkcia vráti (ako reťazec) **aktuálny dátum a čas**.

Tabuľka 2 Použitie niektorých formátovacích parametrov pre funkciu date()

príkaz	vypíše sa
<code>date("j.n.Y")</code>	6.8.2018
<code>date("j. F Y, h:i A")</code>	6. August 2018, 08:24 AM
<code>date("G:i:s")</code>	8:26:19
<code>if (date('G') <= 6) { echo 'Dobré ráno
'; }</code>	
<code>if (date('n') == 2) { echo 'Február
'; }</code>	

Kompletný zoznam znakov, ktoré sú rozpoznávané v reťazcovom parametri `format`, sú v súbore `date-format.html` (prevzaté z manuálu PHP⁷). V prípade, že v reťazci `format` uvedieme znak, ktorý funkcia `date()` nerozpozná, vypíše ho ako obyčajný znak (napr. bodka, dvojbodka, medzera, čiarka, ...).

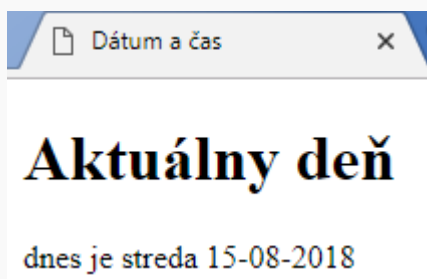
POZNÁMKA

Pri práci s dátumom nemusí mať webserver nastavenú tzv. časovú zónu. V prípade, že časové funkcie nedávajú správne hodnoty, resp. dávajú upozornenie (Warning), nastavíme časovú zónu príkazom `date_default_timezone_set('Europe/Bratislava');` pred použitím príkazu `date()`.



ÚLOHA 4.5

Vypíšte aktuálny dátum v tvare "deň-mesiac-rok", kde deň a mesiac majú vždy dve čísla, napr.: 06-08-2018. K dátumu pripíšte, aký deň v týždni to je, vypíšte to po slovensky. Napr.:



Návod: Môžete využiť formát `w` a pole s názvami dní v správnom poradí.



⁷ www.php.net



ÚLOHA 4.6

Vypíšte aktuálny dátum tak, aby sa mesiac vypísal v slovenčine.

4.3 Vlastné funkcie

Definovanie vlastných funkcií existuje vo väčšine programovacích jazykov a PHP nie je výnimkou. Funkcie nám umožňujú znovu využiť časť kódu pre rovnakú akciu a aj sprehľadňujú a zjednodušujú kód.

Vlastné funkcie definujeme pomocou nasledujúcej štruktúry:

```
function menoFunkcie() {  
    // telo funkcie  
}
```

Názov funkcie by mal byť krátky a výstižný. Nesmie mať rovnaký názov ako už existujúce PHP funkcie. V jej názve môžeme použiť len písmená, číslice a podčiarkovník. Názov funkcie nemôže začínať číslicou.

Funkciu definujeme v rámci PHP kódu, ale skôr, ako ju potrebujeme zavolať.



PRÍKLAD 4.7

04/vypis01.p
hp

Zadefinujme funkciu `vypis()`, ktorá bude vypisovať čísla od 1 do 10 oddelené medzerou. Potom zavolajme funkciu `vypis()`.

```
<?php  
function vypis() {  
    for ($i = 1; $i <= 10; $i++) {  
        echo "$i ";  
    }  
}  
  
vypis();  
?>
```

Všimnite si, že funkciu sme zavolali aj s prázdnyimi zátvorkami. Keby sme ju zapísali bez zátvoriek, tak by ju interpreter považoval za konštantu a ak taká konštantá neexistuje, vypísala by sa chyba.

Parametre funkcií

Väčšina funkcií, ktoré budeme programovať, bude potrebovať parametre. Parametre uvádzame v okrúhlych zátvorkách za názvom funkcie.

```
function menoFunkcie(parametre) {  
    // telo funkcie  
}
```

PRÍKLAD 4.8



04/vypis02.php

Upravme funkciu `vypis()` z predchádzajúceho príkladu tak, aby vypisovala všetky čísla od 1 po zadané číslo. Potom vypíšme všetky čísla od 1 do 15 a do ďalšieho riadku od 1 do 5.

```
function vypis($do) {
    for ($i = 1; $i <= $do; $i++) {
        echo "$i ";
    }
}

vypis(15);
echo "<br>\n";
vypis(5);
```

ÚLOHA 4.9



Napíšte funkciu `privitaj()`, ktorá ako parameter dostane *meno* a vypíše vetu "Vitaj na stránke *meno*". Vytvorte formulár s textovým poľom podobne ako v príklade 3.6. Po odoslaní formulára zavolajte funkciu `privitaj()` so zadaným menom. Výsledná stránka môže vyzeráť nasledovne:

Formulár

Vitaj na stránke Jožko.

Meno:

Odošli údaje

Ak má funkcia viac parametrov, tak ich oddeľujeme čiarkou. Teda zapisujeme:

- `function moja_funkcia($x) { }`
- `function moja_funkcia($x, $y) { }`
- `function moja_funkcia($x, $y, $z) { }`

PRÍKLAD 4.10



04/vypis03.php

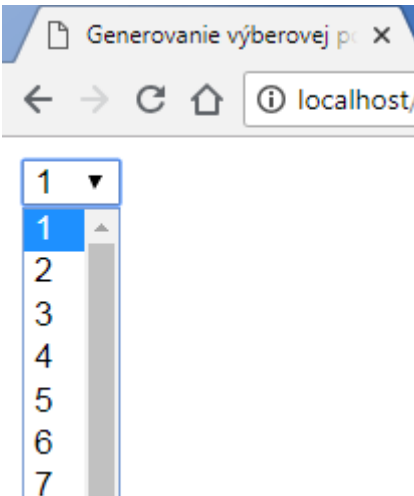
Napíšme funkciu `vypis_poloziiek($od, $do)`, ktorá pre formulárový prvok `<select>` vytvorí jednotlivé číselné položky (`option`), ktoré budú začínať číslom `$od` a končiť číslom `$do`. Potom vytvoríme formulár s prvkom `<select>`, ktorého meno je *den* a položky bude mať od 1 do 31.


```

<?php
function vypis_poloziek($od, $do) {
    for ($i = $od; $i <= $do; $i++) {
        echo "<option value='$i'>$i</option>\n";
    }
}

echo "<form>\n";
echo "<select name='den'>\n";
vypis_poloziek(1,31);
echo "</select>\n";
echo "</form>\n";
?>

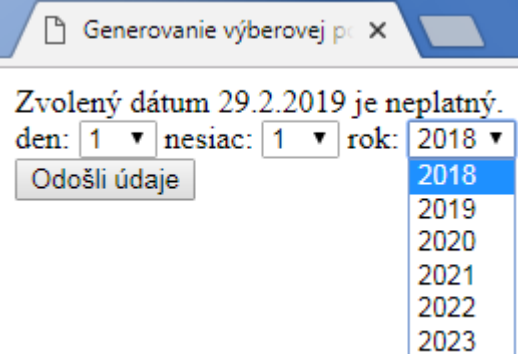
```

zdrojový HTML kód	výstup v prehliadači
<pre> 9 <form> 10 <select name='den'> 11 <option value='1'>1</option> 12 <option value='2'>2</option> 13 <option value='3'>3</option> 14 <option value='4'>4</option> 15 <option value='5'>5</option> ... 39 <option value='29'>29</option> 40 <option value='30'>30</option> 41 <option value='31'>31</option> 42 </select> 43 </form> </pre>	



ÚLOHA 4.11

Napište skript, v ktorom vytvoríte formulár s tromi položkami `<select>` pre dátum, mesiac a rok (rok od aktuálneho roka + 5 ďalších rokov) a odosielacím tlačidlom. Po odoslaní formulára vypíšte vybraný dátum a aj to, či je tento dátum platný. Výsledná stránka môže vyzerat nasledovne:



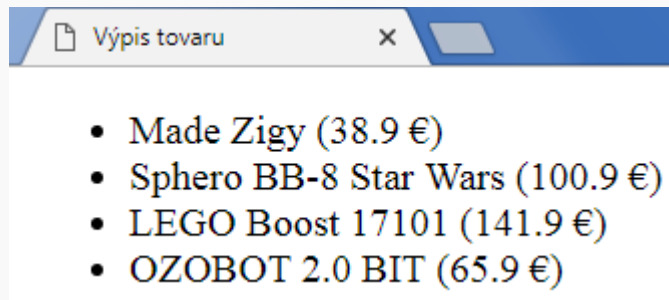
ÚLOHA 4.12

Napíšte funkciu `vypis_tovaru($pole)`, ktorá očakáva nasledujúcu štruktúru vstupného poľa: `['nazov1' => cena1, 'nazov2' => cena2, 'nazov3' => cena3, ...]`.

Funkcia vypíše jednotlivé prvky poľa ako zoznam s odrážkami v tvare "názov tovaru (cena)". Pomocou tejto funkcie vypíšete pole

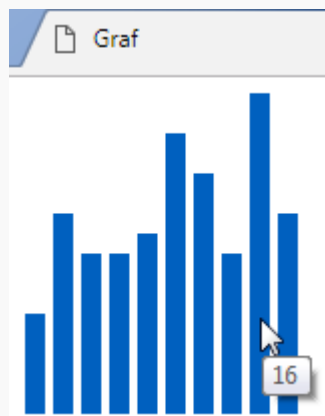
```
$tovar = ['Made Zigy' => 38.90, 'Sphero BB-8 Star Wars' => 100.90, 'LEGO Boost 17101' => 141.90, 'OZOBOT 2.0 BIT' => 65.90];
```

Výsledná stránka by mohla vyzeráť nasledovne:



ÚLOHA 4.13

Napíšte funkciu `stlpcovy_graf()`, ktorá ako parameter dostane číselné pole a hodnoty poľa zobrazí ako stĺpcový graf. Vedľa seba budú obrázky (element ``), ktorých výška bude podľa príslušnej hodnoty z poľa (ak treba môže sa vynásobiť 10). Elementom `` treba nastaviť vlastnosť `title`, ktorá bude reprezentovať danú hodnotu stĺpca. Napr.: pre pole `[5, 10, 8, 8, 9, 14, 12, 8, 16, 10]` sa vytvorí nasledujúci graf:



Funkcie s návratovou hodnotou

Občas by sme potrebovali, aby funkcia nič nevypisovala, ale napríklad niečo vypočítala a vrátila výsledok. PHP na vrátenie hodnoty využíva `return`, podobne ako aj niektoré iné programovacie jazyky.



ZAPAMÄTATEJ SI

Príkazom `return` sa sa vykonávanie funkcie končí, aj keby bol za ním ešte iný príkaz.



PRÍKLAD 4.14

04/mocnina0
3.php

Napišme funkciu `mocnina3()`, ktorá vráti tretiu mocninu zadaného čísla. Potom vypíšme tretie mocniny čísel od 1 do 10.

```
<?php
function mocnina3($cislo) {
    return $cislo * $cislo * $cislo;
}

for ($i = 1; $i <= 10; $i++) {
    echo "$i - " . mocnina3($i) . "<br>\n";
}
?>
```



PRÍKLAD 4.15

04/sucet.php

Napišme funkciu `sucet_cisel($pole)`, ktorá vypočíta súčet čísel v číselnom poli `$pole`. Potom túto funkciu otestujeme na poliach `[2, 1, 4, 3, 6, 5, 8, 7, 10, 9]` a `[2, 3, 5, 7, 11, 13, 17, 19, 23, 29]`.

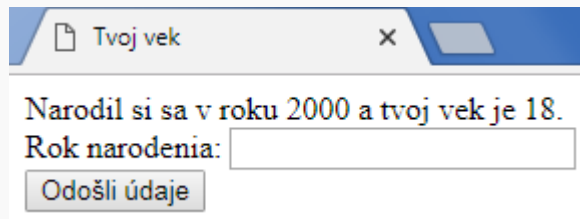
```
<?php
function sucet_cisel($pole) {
    $suc = 0;
    foreach($pole as $hodnota) {
        $suc += $hodnota;
    }
    return $suc;
}

$a = [2, 1, 4, 3, 6, 5, 8, 7, 10, 9];
$b = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29];

print_r($a);
echo "súčet čísel v poli je " . sucet_cisel($a) . "<br>\n";
print_r($b);
echo "súčet čísel v poli je " . sucet_cisel($b) . "<br>\n";
?>
```

ÚLOHA 4.16

Napíšte funkciu `vek()`, ktorá ako parameter dostane rok narodenia a jej výsledkom bude aktuálny vek (vypočíta sa pomocou aktuálneho roku). Vytvorte formulár, v ktorom sa bude dať zadať rok narodenia a po odoslaní formulára sa vypíše príslušný vek. Napr.:

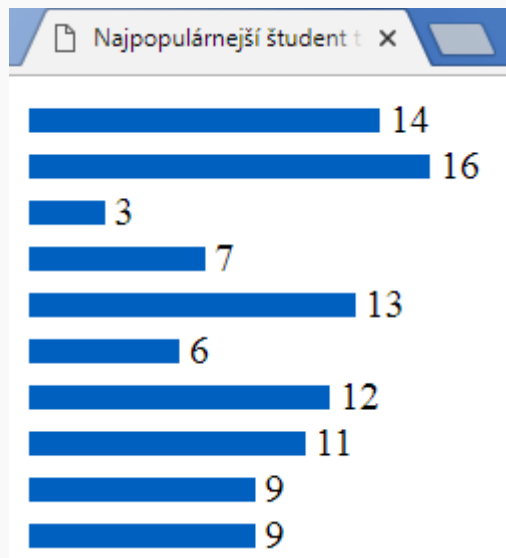


ÚLOHA 4.17

Napíšte funkciu `vytvor($n, $m)`, ktorá bude simulovať hlasovanie za najpopulárnejšieho študenta triedy. Parameter `$n` znamená počet študentov v triede a parameter `$m` znamená počet ľudí, ktorí budú hlasovať. Funkcia najskôr vytvorí pole dĺžky `$n` s hodnotami 0 a potom `$m`-krát vygeneruje náhodné číslo študenta a tomu pripočíta hlas. Výsledkom funkcie bude takto vygenerované pole.

Ďalej vytvorte funkciu `zobraz_vysledky($pole)`, ktorá ako parameter dostane pole vytvorené predchádzajúcou funkciou `vytvor()`. Táto funkcia vytvorí riadkový graf pre jednotlivé prvky poľa. V každom riadku bude jeden obrázok (element ``), ktorého šírka bude podľa príslušnej hodnoty z poľa (odporúčame vynásobiť 10, aby boli rozdiely viditeľnejšie).

Výsledná stránka pre volanie `vytvor(10,100)` a `zobraz_vysledky(pole)` by mohla vyzeráť nasledovne:



4.4 Viaceré PHP vsuvky

Doteraz sme naše stránky riešili tak, že sme všetko, čo sme chceli vygenerovať pomocou PHP, dávali do jednej PHP vsuvky. V rámci stránky však môžeme mať viac PHP vsuviiek kombinovaných s HTML kódom. Tak nemusíme všetky HTML elementy vypisovať pomocou

príkazu `echo`. Ak niečo definujeme v jednej PHP vsuvke, môžeme to v niektorej nasledujúcej PHP vsuvke použiť. Napr.:

04/viacere_vsuvky.php

```
1 <?php
2 $nadpis = 'Moja webstránka';
3 $autor = 'Janko Hraško';
4 ?>
5 <!DOCTYPE html>
6 <html>
7 <head>
8 <meta charset="utf-8">
9 <title><?php echo $nadpis; ?></title>
10 </head>
11
12 <body>
13 <h1><?php echo $nadpis; ?></h1>
14 <footer>vytvoril: <?php echo $autor; ?></footer>
15 </body>
16 </html>
```

V úlohe 3.11 sa mali vo výberovej ponuke vypísať rôzne druhy pizze. Keby sme však mali položiek veľa, tak by sme to určite neriešili písaním všetkých položiek samostatne, ale využili by sme PHP skript na jej vygenerovanie. Potrebujeme však mať údaje uložené v nejakej štruktúre. Vyriešme podobný príklad.



04/vyber_tovaru.php

PRÍKLAD 4.18

Vytvorme výberovú ponuku s robotickými hračkami. Údaje o hračkách máme uložené v poli `$tovar = ['Made Zigy' => 38.90, 'Sphero BB-8 Star Wars' => 100.90, 'LEGO Boost 17101' => 141.90, 'OZOBOT 2.0 BIT' => 65.90];`, ktoré si môžeme zdefinovať na začiatku stránky. Vo výberovej ponuke sa budú zobrazovať názvy aj s cenou, ale po odoslaní formulára sa vypíše iba názov.

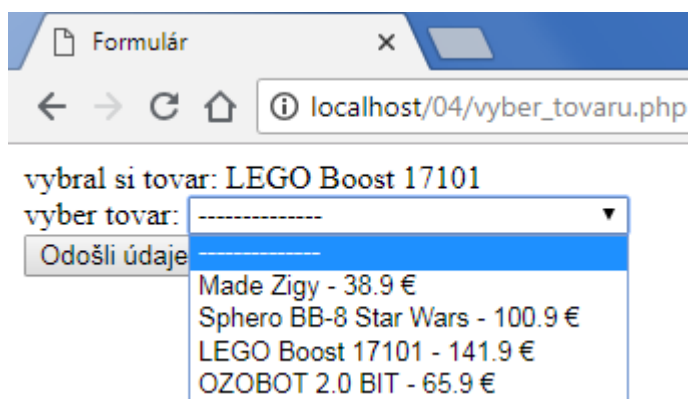
```

1  <?php
2  $tovar = ['Made Zigy' => 38.90, 'Sphero BB-8 Star Wars' => 100.90
3  , 'LEGO Boost 17101' => 141.90, 'OZOBOT 2.0 BIT' => 65.90];
4  ?>
5  <!DOCTYPE html>
6  <html>
7  <head>
8  <meta charset="utf-8">
9  <title>Formulár</title>
10 </head>
11 <body>
12 <?php
13 if (isset($_POST['tlacidlo'])) {
14     if (isset($_POST['vyber']) && ($_POST['vyber'] != '')) {
15         echo 'vybral si tovar: ' . $_POST['vyber'];
16     } else {
17         echo 'Nevybral si žiaden tovar!';
18     }
19 }
20 ?>
21
22 <form method="post">
23     <label for="vyber">vyber tovar:</label>
24     <select name="vyber" id="vyber">
25         <option value="">-----</option>
26     <?php
27     foreach($tovar as $kluc => $hodnota) {
28         echo "<option value='$kluc'>$kluc - $hodnota €</option>";
29     }
30     ?>
31     </select><br>
32     <input type="submit" name="tlacidlo" value="Odošli údaje">
33 </form>
34
35 </body>
36 </html>

```

PHP vsuvky

Výsledný formulár môže vyzerať nasledovne:



PRÍKLAD 4.19

Aby sme mohli vyberať viac tovarov, upravme náš predchádzajúci príklad tak, že namiesto výberovej ponuky použijeme začiarkávacie políčka.

Najskôr upravme len formulár:

```

<form method="post">
<?php
foreach($tovar as $kluc => $hodnota) {
    echo "<label for='\$kluc'>\$kluc - \$hodnota €</label> <input
type='checkbox' name='\$kluc' value='\$kluc' id='\$kluc'><br>";
}
?>
    <input type="submit" name="tlacidlo" value="Odošli údaje">
</form>

```

A teraz overovanie. Keďže potrebujeme overiť všetky začiarkávacie políčka a ich mená (`name`) sú kľúče z nášho poľa, môžeme na to využiť cyklus.

```

if (isset($_POST['tlacidlo'])) {
    echo "vybral si tento tovar: ";
    foreach($tovar as $kluc => $hodnota) {
        if (isset($_POST[$kluc])) {
            echo $_POST[$kluc] . ", ";
        }
    }
    echo "<br>\n";
}

```

Aj napriek tomu, že začiarkneme nejaké produkty, nevypíšu sa nám. Dajme si vypísať `$_POST`.

```

if (isset($_POST['tlacidlo'])) {
    print_r($_POST);
    echo "<br>";
    echo "vybral si tento tovar: ";
    ...
}

```

A zobrazí sa:

Môžeme si všimnúť, že kľúče v tomto poli sa nezhodujú s kľúčmi v poli `$tovar` - medzery sú nahradené podčiarkovníkmi. To znamená, že naše kľúče v poli `$_POST` naozaj nemohol nájsť. Problémom je atribút `name` (`id`) elementu `input` (ale aj iných elementov), ktoré nesmú obsahovať medzeru.

To znamená, že naša štruktúra údajov uložená v poli `$tovar` nie je vyhovujúca. Aby sme mohli aj naďalej generovať formulár a aj jeho overovanie pomocou cyklu, potrebujeme vymyslieť inú štruktúru údajov. Ukážeme rôzne riešenia.

[Viac jednoduchých polí](#)



PRÍKLAD 4.20

Vyriešime príklad 4.19 pomocou dvoch jednoduchých polí.

Nahradíme pole `$tovar = ['Made Zigy' => 38.90, 'Sphero BB-8 Star Wars' => 100.90, 'LEGO Boost 17101' => 141.90, 'OZOBOT 2.0 BIT' => 65.90];` dvoma poľami `$tovar_nazvy = ['Made Zigy', 'Sphero BB-8 Star Wars', 'LEGO Boost 17101', 'OZOBOT 2.0 BIT'];` a `$tovar_ceny = [38.90, 100.90, 141.90, 65.90];`. Obe polia sú rovnako dlhé a cena zodpovedá názvu na rovnakej pozícii.

	0	1	2	3
\$tovar_nazvy	'Made Zigy'	'Sphero BB-8 Star Wars'	'LEGO Boost 17101'	'OZOBOT 2.0 BIT'
\$tovar_ceny	38.90	100.90	141.90	65.90

Teraz upravme formulár:

```
<form method="post">
<?php
foreach($tovar_nazvy as $kluc => $hodnota) {
    echo "<label for='t_{$kluc}'>$hodnota - $tovar_ceny[{$kluc}] €</label>";
    <input type='checkbox' name='t_{$kluc}' value='{$hodnota}'
    id='t_{$kluc}'><br>\n";
}
?>
<input type="submit" name="tlacidlo" value="Odošli údaje">
</form>
```

Stačí nám prechádzať cez jedno pole (lebo indexy v oboch poliach sú rovnaké). Atribút `for` pre element `label` a atribúty `name` a `value` pre element `input` majú rovnakú hodnotu - `t_{$kluc}`. Prečo sme nemohli dať len `$kluc`? Lebo `$kluc` nadobúda číselné hodnoty a vieme, že `name` a `id` nesmú začínať číslom. Tak sme preto pridali pred `$kluc` prefix `t_`. Text v `label` sme skombinovali z oboch polí ako hodnoty z poľa, ktoré prechádzame (`$tovar_nazvy`) a hodnoty z poľa `$tovar_ceny` na rovnakom indexe ako je kľúč v prechádzanom poli.

Dokončíme overovanie. Najskôr jednoduchšia verzia:

```
if (isset($_POST['tlacidlo'])) {
    echo "vybral si tento tovar: ";
    foreach($tovar_nazvy as $kluc => $hodnota) {
        if (isset($_POST["t_{$kluc}"])) {
            echo $_POST["t_{$kluc}"] . ", ";
        }
    }
    echo "<br>\n";
}
```

Toto riešenie už funguje, ale môže byť neefektívne (keď je pole tovarov väčšie ako pole `$_POST`). Prechádzame cez celé pole `$tovar_nazvy`, aj keď v poli `$_POST` nebude žiaden tovar.

Môžeme prechádzať cez pole `$_POST`. Z tohto poľa nás budú zaujímať len tie prvky, ktorých kľúče začínajú `t_`. Na to využijeme reťazcovú funkciu `strpos()`.

04/vyber_tovaru2
0a.php

04/vyber_tovaru
20b.php

Funkcia `strpos(reťazec1, reťazec2)` vráti pozíciu prvého výskytu `reťazec2` v `reťazec1` alebo `FALSE`, ak sa `reťazec2` v `reťazec1` nenašiel. Záleží na veľkých a malých písmenách. Napr.:

príkaz	výsledok
<code>strpos('drevo', 'rev')</code>	1
<code>strpos('drevo', 'lev')</code>	FALSE
<code>strpos('drevo', 'd')</code>	0
<code>strpos('drevo', 'D')</code>	FALSE

Potrebuje teda zistiť: `if (strpos($kluc, 't_') == 0)`

Vyskúšajme túto podmienku na inom príklade:

```
if (strpos('Ahoj', 'A') == 0) {  
    echo "Nachádza<br>\n";  
} else {  
    echo "Nenachádza<br>\n";  
}  
if (strpos('Ahoj', 'a') == 0) {  
    echo "Nachádza<br>\n";  
} else {  
    echo "Nenachádza<br>\n";  
}
```

Predpokladali by sme, že v prvom prípade to vypíše *Nachádza* a v druhom *Nenachádza*, ale nám to v oboch prípadoch vypíše *Nachádza*. Prečo sa druhý prípad zle vyhodnotil?

Funkcia `strpos('Ahoj', 'a')` vráti `FALSE` a pri porovnaní pomocou `==` ho PHP pretypuje na `0` (písali sme o tom aj v 2. kapitole). Vyriešime to pomocou operátora `===`, ktorý kontroluje rovnosť hodnôt aj typu, teda či sú hodnoty `0` a `FALSE` nie sú identické.



ÚLOHA 4.21

Otestujte predchádzajúci kód s opraveným operátorom.

Aby sme mohli zisťovať údaje z poľa `$tovar_nazvy`, potrebujeme z kľúča (ktorý získame z poľa `$_POST`) odstrániť `t_`. Na to využijeme funkciu `substr()`.

Funkcia `substr(reťazec, štart, dĺžka)` vráti časť reťazca. `Reťazec` a `štart` sú povinné parametre. Parameter `štart` určuje, na akej pozícii začína výsledný reťazec. `0` znamená prvý znak v reťazci. Parameter `dĺžka` je voliteľný a znamená koľko maximálne znakov bude mať výsledný reťazec. Ak ho neuviedeme, tak výsledkom budú všetky znaky od štart po koniec reťazca. Napr.: `$vstup = "PHP v príkladoch"`

príkaz	vypíše sa
<code>substr(\$vstup, 4)</code>	v príkladoch
<code>substr(\$vstup, 2, 5)</code>	P v p

Overovanie, v ktorom prechádzame cez pole `$_POST`, môže vyzeráť nasledovne:

```
if (isset($_POST['tlacidlo'])) {
    echo "vybral si tento tovar: ";
    foreach($_POST as $kluc => $hodnota) {
        if (strpos($kluc, 't_') === 0) {
            $kluc = substr($kluc, 2);
            echo "{$stovar_nazvy[$kluc]}, ";
        }
    }
    echo "<br>\n";
}
```

Viac asociatívnych polí

Aby sme sa vyhli upravovaniu kľúčov, môžeme využiť viac asociatívnych polí. Len musíme dobre definovať kľúče, čo znamená, že nesmú začínať číslom.

PRÍKLAD 4.22

Vyriešme príklad 4.19 pomocou dvoch asociatívnych polí.

Nahradíme pole `$stovar = ['Made Zigy' => 38.90, 'Sphero BB-8 Star Wars' => 100.90, 'LEGO Boost 17101' => 141.90, 'OZOBOT 2.0 BIT' => 65.90];` dvoma poliami `$stovar_nazvy = ['MZ' => 'Made Zigy', 'SSW' => 'Sphero BB-8 Star Wars', 'LB' => 'LEGO Boost 17101', 'OB' => 'OZOBOT 2.0 BIT'];` a `$stovar_ceny = ['MZ' => 38.90, 'SSW' => 100.90, 'LB' => 141.90, 'OB' => 65.90];`.

	'MZ'	'SSW'	'LB'	'OB'
\$stovar_nazvy	'Made Zigy'	'Sphero BB-8 Star Wars'	'LEGO Boost 17101'	'OZOBOT 2.0 BIT'
\$stovar_ceny	38.90	100.90	141.90	65.90

Upravme formulár:

```
<form method="post">
<?php
foreach($stovar_nazvy as $kluc => $hodnota) {
    echo "<label for='$kluc'>$hodnota - $stovar_ceny[$kluc] €</label>";
    <input type='checkbox' name='$kluc' value='$hodnota'
    id='$kluc'><br>\n";
}
?>
<input type="submit" name="tlacidlo" value="Odošli údaje">
</form>
```

V tomto prípade máme kľúče vhodné, preto ich nepotrebujeme nijako upravovať.

04/vyber_tovaru22
a.php

Dorobme overovanie, najskôr cez prechádzanie poľa `$tovar_nazvy`:

```
if (isset($_POST['tlacidlo'])) {
    echo "vybral si tento tovar: ";
    foreach($tovar_nazvy as $kluc => $hodnota) {
        if (isset($_POST[$kluc])) {
            echo $_POST[$kluc] . ", ";
        }
    }
    echo "<br>\n";
}
```

04/vyber_tovaru22
b.php

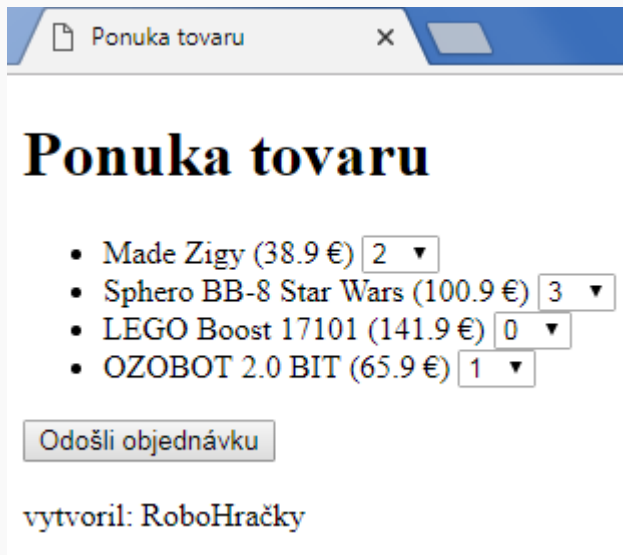
A teraz cez prechádzanie poľa `$_POST`:

```
if (isset($_POST['tlacidlo'])) {
    echo "vybral si tento tovar: ";
    foreach($_POST as $kluc => $hodnota) {
        if (isset($tovar_nazvy[$kluc])) {
            echo $tovar_nazvy[$kluc] . ", ";
        }
    }
    echo "<br>\n";
}
```



ÚLOHA 4.23

Vytvorte stránku ako na obrázku:



Na začiatku stránky v PHP vsuvke zadefinujte premenné `$nadpis`, `$autor` a `$tovar` (alebo `$tovar_nazvy` a `$tovar_ceny`) a funkciu `vypis_poloziok($od, $do)`. V ďalších PHP vsuvkách vypíšte správne titulok stránky, nadpis stránky a päťu stránky. V tele stránky budú v odrážkovom zozname vypísané jednotlivé položky a pri každej bude výberová ponuka (od 0 do 10). Mená jednotlivých výberových ponúk sú kódy tovarov. Po odoslaní formulára sa vypíše rekapitulácia objednávky ako na obrázku:

Rekapitulácia objednávky:

Made Zigy: 2 ks

Sphero BB-8 Star Wars: 3 ks

OZOBOT 2.0 BIT: 1 ks

Ponuka tovaru

- Made Zigy (38.9 €) ▼

Otestuje predchádzajúci kód s opraveným operátorom.

4.5 Metodické pokyny pre učiteľa



CIEĽ

Doplniť základné informácie o jazyku PHP - o prácu s náhodnými číslami, dátumom a časom, o vytváraní vlastných funkcií a o využívaní viacerých PHP vsuviiek.



VÝKLAD

Táto časť je zameraná na:

- náhodné čísla,
- dátum a čas,
- vytváranie vlastných funkcií,
- využívanie viacerých PHP vsuviiek.

Popri tom treba študentom vysvetliť niektoré reťazcové funkcie a operátor rovnosti hodnôt aj typov.

Výklad sa strieda s riešenými príkladmi a samostatnými úlohami pre študentov.

Náhodné čísla

`getrandmax()` je funkcia, ktorá vracia najväčšiu (celočíselnú) možnú hodnotu, ktorá môže byť vygenerovaná funkciou `rand()`

Dátum a čas

USB Webserver nemá (v niektorých verziách), bohužiaľ, štandardne nastavenú časovú zónu, preto sa nedá správne určiť čas a pracovať s ním. Ak nechceme študentov zaťažovať nastavovaním zóny pomocou príkazu `date_default_timezone_set()`, môžeme zmeniť nastavenie USB Webservera - v súbore `settings/php.ini` odkomentujeme (zrušíme bodkočiarku pred) príkaz `date.timezone = Europe/Paris`.

`date()` - funkcia má aj druhý nepovinný parameter timestamp, ktorý reprezentuje zadaný dátum a čas v špeciálnom tvare (vieme ho získať pomocou funkcie `mktime(hodiny, minúty, sekundy, mesiac, deň, rok)`)

Vlastné funkcie

Jazyk PHP nepozná pojem procedúra a teda všetky vlastné príkazy sú funkcie. V prípade, že chceme, aby funkcia skutočne vrátila nejakú hodnotu, použijeme príkaz `return`. Vlastná funkcia nesmie mať rovnaký názov ako už existujúce PHP funkcie a ani iné vlastné funkcie, lebo PHP nepodporuje prekrývanie funkcií.

K úlohe 4.13: Úloha by sa dala rozšíriť tak, že by nebol graf jednofarebný, ale použilo by sa viacero farieb (farebných obrázkov) a precvičili by sa náhodné čísla. Farby (názvy súborov) by mohli byť uložené v poli, napr. `$farby = ['modra.gif', 'cervena.gif', 'zlta.gif,`

'zelena.gif', 'cierna.gif']. Podľa dĺžky poľa by sa vygenerovalo číslo od 1 do dĺžky poľa a príslušná farba by sa použila v grafe.

Ešte rozšírený variant úlohy by mohol byť, že farby by boli ako položky výberovej ponuky a po odoslaní formulára by sa graf vykreslil príslušnou farbou.

Parametre – inicializačná hodnota

Toto je rozširujúca časť. Je na zvážení učiteľa, či sa jej bude so študentmi venovať.

PRÍKLAD

Upravme funkciu `vypis_poloziiek()` z príkladu 4.10 tak, aby vypísala všetky čísla od čísla `$od` po číslo `$do` s krokom `$krok`.



02/vypis05a.ph
P,
02/vypis05b.ph
P

Funkcia by mohla vyzerať nasledovne:

```
<?php
function vypis_poloziiek($od, $do, $krok) {
    for ($i = $od; $i <= $do; $i += $krok) {
        echo "<option value='$i'>$i</option>\n";
    }
}
```

A volať by sme ju mohli napríklad takto:

```
vypis_poloziiek(1,12,1);
vypis_poloziiek(2,20,2);
vypis_poloziiek(4,40,4)
```

Ak by sme väčšinou potrebovali vypisovať čísla s krokom 1 a len vo výnimočných prípadoch chceme výpis s iným krokom, hodilo by sa nám, keby sme funkciu `vypis_poloziiek()` volali len s dvomi parametrami a vo výnimočných prípadoch by sme funkciu zavolali so všetkými tromi. Presne takúto funkčnosť nám umožňuje tzv. **inicializačná hodnota parametra funkcie** – parametru funkcie priradíme hodnotu. V prípade, že taká hodnota je na konci zoznamu parametrov, nemusíme parameter vôbec uvádzať, a teda počet parametrov bude menší.

Uvedme si to však na našom príklade funkcie `vypis_poloziiek()`, ktorú by sme mohli zmeniť nasledovne:

```
<?php
function vypis_poloziiek($od, $do, $krok = 1 ) {
    for ($i = $od; $i <= $do; $i += $krok) {
        echo "<option value='$i'>$i</option>\n";
    }
}
```

Potom môžeme takto definovanú funkciu zavolať aj uvedenými spôsobmi:

```
vypis_poloziiek(2018,2030);
vypis_poloziiek(1,28);
vypis_poloziiek(2,10,2)
```

Našu funkciu by sme mohli upraviť aj takto:

```
function vypis_poloziiek($od = 1, $do = 10, $krok = 1) {}
```

Všetkým trom parametrom sme dali inicializačné hodnoty.



ÚLOHA

Zamyslite sa a odpovedzte, aké čísla sa zobrazia vo výberovej ponuke pri rôznych volaniach funkcie `vypis_poloziiek($od=1, $do=10, $krok=1)`:

- `vypis_poloziiek(0,30,3);`
- `vypis_poloziiek(1,50,5);`
- `vypis_poloziiek(1,31);`
- `vypis_poloziiek(11,20);`
- `vypis_poloziiek(5);`
- `vypis_poloziiek(10);`
- `vypis_poloziiek();`

Viaceré PHP vsuvky

K úlohe 4.19: V učebnici sú uvedené 2 spôsoby pomocou rôznych polí. Učiteľ môže zvážiť, či ukáže obe, alebo len jedno z nich. Táto úloha by sa dala vyriešiť aj pomocou jednej zložitejšej štruktúry.

Jedna zložitejšia štruktúra

Posledná možnosť, ktorú si ukážeme, využíva iba jednu zložitejšiu štruktúru. Každý tovar bude jedno asociatívne pole s kľúčmi `'kod'`, `'nazov'` a `'cena'`. Potom tieto asociatívne polia spojíme do jedného veľkého poľa. Tento spôsob sa najviac približuje spracovaniu údajov z databáz (tomu sa však už nebudeme venovať).



04/vyber_tov
aru24.php

PRÍKLAD

Vyriešime príklad 4.19 pomocou údajov z jednej štruktúry.

Nahradíme pôvodné pole `$tovar` nasledujúcou štruktúrou:

```
$tovar = [['kod' => 'MZ', 'nazov' => 'Made Zigy', 'cena' => 38.90],  
          ['kod' => 'SSW', 'nazov' => 'Sphero BB-8 Star Wars',  
           'cena' => 100.90],  
          ['kod' => 'LB', 'nazov' => 'LEGO Boost 17101',  
           'cena' => 141.90],  
          ['kod' => 'OB', 'nazov' => 'OZOBOT 2.0 BIT',  
           'cena' => 65.90]];
```

Zodpovedá jej nasledujúci obrázok:

	'kod'	'nazov'	'cena'
0	'MZ'	'Made Zigy'	38.90
1	'SSW'	'Sphero BB-8 Star Wars'	100.90
2	'LB'	'LEGO Boost 17101'	141.90
3	'OB'	'OZOBOT 2.0 BIT'	65.90

Každý prvok nášho poľa `$tovar` je vlastne asociatívne pole.

Skúsme vypísať niektoré prvky poľa `$tovar`:

<code>print_r(\$tovar[0]);</code>	Array ([kod] => MZ [nazov] => Made Zigy [cena] => 38.9)
<code>print_r(\$tovar[2]);</code>	Array ([kod] => LB [nazov] => LEGO Boost 17101 [cena] => 141.9)

Ak chceme vypísať názov tovaru s indexom 1, tak zadáme `echo $tovar[1]['nazov']` a pod.

Upravme najskôr formulár:

```
<form method="post">
<?php
foreach($tovar as $hodnota) {
    echo "<label for='{ $hodnota['kod'] }'>{ $hodnota['nazov'] } -
{ $hodnota['cena'] } €</label> <input type='checkbox'
name='{ $hodnota['kod'] }' value='{ $hodnota['nazov'] }'
id='{ $hodnota['kod'] }'><br>\n";
}
?>
<input type="submit" name="tlacidlo" value="Odošli údaje">
</form>
```

Uvedomme si, že v `$hodnota` sú postupne jednotlivé asociatívne polia, ako prvé - `['kod' => 'MZ', 'nazov' => 'Made Zigy', 'cena' => 38.90]`.

A nakoniec dokončíme overovanie. Spravíme to len cez prechádzanie poľom `$tovar`.

```
if (isset($_POST['tlacidlo'])) {
    echo "vybral si tento tovar: ";
    foreach($tovar as $hodnota) {
        if (isset($_POST[$hodnota['kod']])) {
            echo $_POST[$hodnota['kod']] . ", ";
        }
    }
    echo "<br>\n";
}
```


Užitočné reťazcové funkcie

názov funkcie	popis
<code>strlen(ret)</code>	Vráti dĺžku reťazca.
<code>substr(ret, zac [, pocet])</code>	Vráti podreťazec dĺžky pocet od znaku na pozícii zac.
<code>strtolower(ret)</code>	Vráti reťazec prevedený na malé písmená.
<code>strtoupper(ret)</code>	Vráti reťazec prevedený na veľké písmená.
<code>strrev(ret)</code>	Vráti prevrátený reťazec.
<code>trim(ret), ltrim(ret), rtrim(ret)</code>	Odstráni tzv. whitespace znaky z príslušného konca.
<code>chr(ascii-kod)</code>	Vráti znak zodpovedajúci ascii-kódu.
<code>ord(reťazec)</code>	Vráti ASCII kód prvého znaku reťazca.
<code>nl2br()</code>	Prevedie konce riadkov (\n) na .
<code>strpos(kde, čo [, zac])</code>	Vráti prvú pozíciu čo v reťazci kde zľava alebo FALSE.
<code>strrpos(kde, čo [, zac])</code>	Vráti prvú pozíciu čo (len 1 znak) v reťazci kde zprava, alebo FALSE.
<code>strrchr(kde, čo)</code>	Vráti podreťazec od pozície čo v reťazci kde zprava do konca reťazca, alebo FALSE. Vyhľadáva len znaky.
<code>strstr(kde, čo)</code>	Vráti podreťazec od pozície čo v reťazci kde zľava do konca reťazca, alebo FALSE.
<code>str_replace(čo, čím, kde)</code>	Nahradí všetky výskyty čo reťazcom čím v kde.

Funkcia `substr(reťazec, štart, dĺžka)` Môže mať aj záporné parametre. Záporná hodnota `štart` definuje pozíciu od konca reťazca. Záporný parameter `dĺžka` určuje, koľko znakov od konca nebude vo výslednom reťazci. Napr.: `$vstup = "PHP v príkladoch"`

príkaz	vypíše sa
<code>substr(\$vstup, -3)</code>	och
<code>substr(\$vstup, 4, -3)</code>	v príklad
<code>substr(\$vstup, -7, 4)</code>	klad

5 PHP - SPRACOVANIE FORMULÁROV (2)

V kapitole č. 3 sme si ukázali základné spracovanie formulárov. To má však viacero nedostatkov, ktoré vyriešime v tejto kapitole.

5.1 Zobrazenie odoslaných údajov znovu vo formulári

DISKUTUJTE

Máme formulár, v ktorom sú niektoré položky povinné. Čo sa má stať, keď nevyplníme všetky povinné položky a odošleme formulár?

Ako by sa mal znovu zobraziť takto odoslaný formulár?



Ak po odoslaní formulára ho znovu zobrazíme a chceme v ňom zobraziť už odoslané údaje, musíme nastaviť tzv. default hodnoty pre jednotlivé HTML elementy. Toto nastavenie sa líši v závislosti od typu prvku formulára - rôzne prvky majú iný spôsob. Poďme si ich postupne ukázať a vysvetliť.

Jednoriadkové textové pole

Ak chceme v textovom poli zobraziť nejakú hodnotu pri načítaní formulára (prednastavená hodnota), musíme nastaviť atribút `value`.

```
<input type="text" name="meno" id="meno" value="hodnota">
```

rovnako by sme pracovali aj s prvkom pre heslo (typ password)

PRÍKLAD 5.1

Doplňme formulár z príkladu 3.6 (03/formular06.php) tak, aby sa odoslané meno zobrazilo vo formulárovom poli, teda do atribútu `value` dáme hodnotu z poľa `$_POST`.



05/formular01.php

```
<label for="meno">Meno:</label> <input type="text" name="meno" id="meno" value="<?php if (isset($_POST['meno'])) {echo $_POST['meno'];} ?>">
```

Skôr, ako vypíšeme hodnotu z poľa `$_POST`, musíme overiť, či existuje.

Výsledný formulár po odoslaní môže vyzeráť nasledovne:



ÚLOHA 5.2

Upravte úlohu 3.8 tak, aby sa po odoslaní formulára zobrazili odoslané hodnoty znovu vo formulárových poliach pre meno, priezvisko a mesto.

Viacriadkové textové pole TEXTAREA

Element `<textarea>` nemá žiadny atribút, ktorý treba nastaviť, ak v ňom chceme zobrazíť nejakú hodnotu. Takáto default hodnota sa vkladá medzi otváraciu a zatváraciu značku – medzi nimi by nemali byť žiadne medzery ani nové riadky.

```
<textarea name="adresa">hodnota</textarea>
```



05/formular02.php

PRÍKLAD 5.3

Doplňte formulár z príkladu 3.7 (03/formular07.php) tak, aby sa odoslaná adresa zobrazila vo formulárovom poli, teda medzi otváraciu a koncovú značku dáme hodnotu z poľa `$_POST`.

```
<textarea name="adresa" id="adresa"><?php if (isset($_POST['adresa'])) {echo $_POST['adresa'];} ?></textarea>
```

Výsledný formulár po odoslaní môže vyzerať nasledovne:



ÚLOHA 5.4

Upravte úlohu 5.2 tak, že prvok mesto bude element `<textarea>`. Zachovajte zobrazovanie všetkých odoslaných hodnôt vo formulári.

Výberová ponuka SELECT

Ak chceme označiť (prednastaviť) niektorú položku výberovej ponuky, nenastavujeme to elementu `<select>`, lebo ten nemá žiaden vhodný atribút. Musíme konkrétnej položke (element `<option>`), zadať atribút `selected`.

```
<select name="pocet" id="pocet">
  <option value="">0</option>
  <option value="1">1</option>
  <option value="2" selected>2</option>
  <option value="3">3</option>
  <option value="4">4</option>
</select>
```

PRÍKLAD 5.5

V elemente `<select>` máme možnosť výberu štyroch čísel. Doplňme do formulára kód tak, aby sa zobrazila zvolená hodnota aj po odoslaní formulára. (Mierne upravíme formulár z príkladu 3.10.).

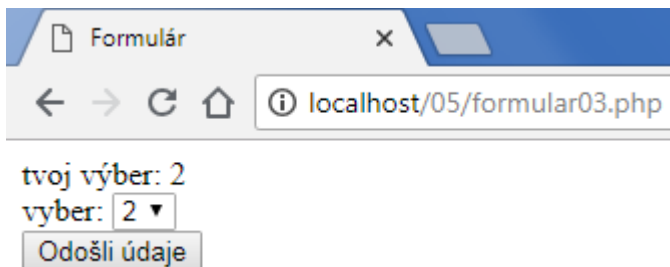


05/formular03.
php

```
<select name="pocet" id="pocet">
  <option value="">0</option>
  <option value="1"
  <?php if (isset($_POST['pocet']) && ($_POST['pocet'] == 1)) {
    echo 'selected';} ?>
  >1</option>
  <option value="2"
  <?php if (isset($_POST['pocet']) && ($_POST['pocet'] == 2)) {
    echo 'selected';} ?>
  >2</option>
  <option value="3"
  <?php if (isset($_POST['pocet']) && ($_POST['pocet'] == 3)) {
    echo 'selected';} ?>
  >3</option>
  <option value="4"
  <?php if (isset($_POST['pocet']) && ($_POST['pocet'] == 4)) {
    echo 'selected';} ?>
  >4</option>
</select>
```

Do každej položky `<option>` musíme vložiť PHP kód, ktorý skontroluje, či existuje premenná `$_POST["pocet"]` a či sa rovná hodnote príslušnej položky. Ak áno, tak sa pri nej vypíše `selected`.

Výsledný formulár po odoslaní môže vyzerať nasledovne:



Vidíme, že zobrazenie zvolenej položky pre element `<select>` je zdĺhavé a náchylné na chyby. Je výhodné, ak môžeme jednotlivé položky pre výberovú ponuku vygenerovať pomocou funkcie. Ukážeme si to na funkcii `vypis_poloziiek()`, ktorá nám generuje číselné položky.



05/formular04.php

PRÍKLAD 5.6

Upravme funkciu `vypis_poloziiek()` z príkladu 4.10 tak, aby vedela označiť niektorú z položiek.

Docielime to tak, že funkcii pridáme tretí parameter, ktorý bude reprezentovať položku, ktorú chceme mať označenú.

Doplnená funkcia:

```
function vypis_poloziiek($od, $do, $oznac) {
    for ($i = $od; $i <= $do; $i++) {
        echo "<option value='$i'";
        if ($i == $oznac) {
            echo ' selected';
        }
        echo ">$i</option>\n";
    }
}
```

A zavolanie funkcie:

```
<select name="pocet" id="pocet">
    <option value="">0</option>
<?php
if (isset($_POST['pocet'])) {
    vypis_poloziiek(1, 4, $_POST['pocet']);
} else {
    vypis_poloziiek(1, 4, 0);
}
?>
</select>
```



ÚLOHA 5.7

Upravte úlohu 3.11 tak, aby sa po odoslaní formulára zobrazila zvolená pizza vo výberovej ponuke.

Prepínač - RADIOBUTTON

Ak chceme označiť (prednastaviť) niektorý prepínač, musíme mu zadať atribút `checked`.

```
<input type="radio" name="pocet" value="1" id="pocet1"><br>
<input type="radio" name="pocet" value="2" id="pocet2"><br>
<input type="radio" name="pocet" value="3" id="pocet3" checked><br>
<input type="radio" name="pocet" value="4" id="pocet4"><br>
```

PRÍKLAD 5.8

Máme štyri prepínače, pomocou ktorých môžeme vybrať jedno zo štyroch čísel. Doplňme do formulára kód tak, aby sa zobrazila zvolená hodnota aj po odoslaní formulára. (Mierne upravíme formulár z príkladu 3.12.).



05/formular05.php

```
<label for="pocet1">1</label>
<input type="radio" name="pocet" value="1" id="pocet1"
<?php if (isset($_POST['pocet']) && ($_POST['pocet'] == 1)) {
    echo ' checked';} ?>><br>

<label for="pocet2">2</label>
<input type="radio" name="pocet" value="2" id="pocet2"
<?php if (isset($_POST['pocet']) && ($_POST['pocet'] == 2)) {
    echo ' checked';} ?>><br>

<label for="pocet3">3</label>
<input type="radio" name="pocet" value="3" id="pocet3"
<?php if (isset($_POST['pocet']) && ($_POST['pocet'] == 3)) {
    echo ' checked';} ?>><br>

<label for="pocet4">4</label>
<input type="radio" name="pocet" value="4" id="pocet4"
<?php if (isset($_POST['pocet']) && ($_POST['pocet'] == 4)) {
    echo ' checked';} ?>><br>
```

Do každého prepínača musíme vložiť PHP kód, ktorý skontroluje, či existuje premenná `$_POST["pocet"]` a či sa rovná hodnote príslušného prepínača. Ak áno, tak sa mu pridá atribút `checked`.

Výsledný formulár po odoslaní môže vyzeráť nasledovne:

tvoj výber: 3

1

2

3

4

Odošli údaje



ÚLOHA 5.10

Upravte predchádzajúci príklad tak, aby prepínače generovala funkcia.



ÚLOHA 5.11

Upravte úlohu 3.13 tak, aby po odoslaní formulára bola označená zvolená pizza.

Začiarkávacie políčko - CHECKBOX

Ak chceme označiť (prednastaviť) niektoré začiarkávacie políčka, musíme im zadať atribút `checked`.

```

<input type="checkbox" name="vyber1" value="1" id="vyber1">
<input type="checkbox" name="vyber2" value="2" id="vyber2" checked>
<input type="checkbox" name="vyber3" value="3" id="vyber3" checked>
<input type="checkbox" name="vyber4" value="4" id="vyber4">

```

Pripomeňme si, že začiarkávacích políčok môže byť označených aj viac.



PRÍKLAD 5.12

05/formular0
6.php

Máme štyri začiarkávacie políčka, pomocou ktorých môžeme vybrať niektoré (aj viaceré) zo štyroch čísel. Doplňme do formulára kód tak, aby sa zobrazili zvolené hodnoty aj po odoslaní formulára. (Mierne upravíme formulár z príkladu 3.15.).

```

<label for="vyber1">1</label>
<input type="checkbox" name="vyber1" value="1" id="vyber1"
<?php if (isset($_POST['vyber1'])) {echo ' checked';} ?>
><br>

<label for="vyber2">2</label>
<input type="checkbox" name="vyber2" value="2" id="vyber2"
<?php if (isset($_POST['vyber2'])) {echo ' checked';} ?>
><br>

<label for="vyber3">3</label>
<input type="checkbox" name="vyber3" value="3" id="vyber3"
<?php if (isset($_POST['vyber3'])) {echo ' checked';} ?>
><br>

<label for="vyber4">4</label>
<input type="checkbox" name="vyber4" value="4" id="vyber4"
<?php if (isset($_POST['vyber4'])) {echo ' checked';} ?>
><br>

```

Do každého začiarkávacieho políčka musíme vložiť PHP kód, ktorý skontroluje, či existuje príslušná premenná `$_POST["vyberX"]`. Ak áno, tak sa mu pridá atribút `checked`.

Výsledný formulár po odoslaní môže vyzeráť nasledovne:

ÚLOHA 5.13

Upravte predchádzajúci príklad tak, aby začiarkávacie políčka generovala funkcia.

ÚLOHA 5.14

Upravte úlohu 3.16 tak, aby po odoslaní formulára boli označené zvolené programovacie jazyky.



5.2 Skrývanie formulára

Doteraz sa nám formulár zobrazoval vždy, aj po odoslaní vyplneného formulára. Ale sú situácie, keď to nie je žiadúce. Napr. keď odošleme registráciu, nepotrebujeme znovu vidieť formulár (s vyplnenými hodnotami), skôr očakávame potvrdenie registrácie. Alebo pri vyplňaní testu cez webový formulár tiež nečakáme, že sa nám znovu zobrazí formulár, ale skôr výsledný počet bodov, resp. správnych odpovedí.

PRÍKLAD 5.15

Majme formulár s položkami *meno*, *adresa*, *tovar* (príklady 5.1, 5.3, 4.18), pričom použijeme dve polia s názvami a cenami tovarov (`$tovar_nazvy`, `$tovar_ceny` z príkladu 4.22). Ak budú po odoslaní formulára zadané všetky 3 hodnoty, zobrazí sa rekapitulácia objednávky, ale formulár sa už nezobrazí. Ak nebude niektorá z hodnôt zadaná, zobrazíme upozornenie, že neboli vyplnené všetky údaje a zobrazíme formulár, v ktorom už budú predvyplnené zadané údaje.



05/formular07-zdroj.php

Využijeme kódy z uvedených príkladov, ktoré však spojíme. Upravený zdrojový kód je v súbore 05/formular07-zdroj.php.

Formulár potrebujeme zobrazíť len v dvoch prípadoch:

1. keď prvýkrát zobrazujeme stránku (nebolo odoslané tlačidlo),
2. keď bol odoslaný nekompletne vyplnený formulár.

riešenie je v
05/formular07.p
hp

V zdrojovom kóde sme doplnili komentáre, kde by sme potrebovali zobrazíť ten istý formulár. Mohli by sme celý formulár nakopírovať na obidve miesta a všetko by fungovalo. Ale to nie je veľmi efektívne riešenie. Kód by sme mali mať napísaný iba raz a len ho nejakou spôsobom využívať.

```
if (isset($_POST['tlacidlo'])) {
    if (isset($_POST['meno']) && ($_POST['meno'] != '') &&
        isset($_POST['adresa']) && ($_POST['adresa'] != '') &&
        isset($_POST['vyber']) && ($_POST['vyber'] != '')) {
        // boli odoslané všetky údaje a zobrazíme rekapituláciu
        echo "meno: {"$_POST['meno']}<br>\n";
        echo "adresa: {"$_POST['adresa']}<br>\n";
        echo "tovar: {"$tovar_nazvy[$_POST['vyber']]}<br>\n";
        // tu formulár nezobrazíme
    } else {
        echo 'Nezadal si niektorú položku!<br>';
        // tu treba zobrazíť formulár
    }
} else {
    // tu treba zobrazíť formulár
}
```

Riešením môže byť použitie premennej (napr. `$zobraz_form`), ktorá keď bude `TRUE`, tak kód s formulárom zobrazíme, keď bude `FALSE`, tak formulár nezobrazíme. Pred testom stlačenia odosielacieho tlačidla premennú `$zobraz_form` nastavíme na `TRUE`. A pri teste kompletne vyplneného formulára ju nastavíme na `FALSE`.



ÚLOHA 5.16

Upravte úlohu 4.23 tak, že keď používateľ zadá niektorý z tovarov (nenulový počet), tak sa už formulár znovu nezobrazí. Inak sa formulár bude zobrazovať.

```

9  <?php
10 $tovar_nazvy = ['MZ' => 'Made Zigy', 'SSW' => 'Sphero BB-8 Star
    Wars', 'LB' => 'LEGO Boost 17101', 'OB' => 'OZOBOT 2.0 BIT'];
11 $tovar_ceny = ['MZ' => 38.90, 'SSW' => 100.90, 'LB' => 141.90,
    'OB' => 65.90];
12
13 function vypis_tovarov($nazvy, $ceny, $oznac) {
20
21     $zobraz_form = true;
22     if (isset($_POST['tlacidlo'])) {
23         if (isset($_POST['meno']) && ($_POST['meno'] != '') &&
24             isset($_POST['adresa']) && ($_POST['adresa'] != '') &&
25             isset($_POST['vyber']) && ($_POST['vyber'] != '')) {
26             // boli odoslané všetky údaje a zobrazíme rekapituláciu
27             echo "meno: {$_POST['meno']}<br>\n";
28             echo "adresa: {$_POST['adresa']}<br>\n";
29             echo "tovar: {$tovar_nazvy[$_POST['vyber']]}<br>\n";
30             // tu formulár nezobrazíme
31             $zobraz_form = false;
32         } else {
33             echo 'Nezadal si niektorú položku!<br>';
34             // tu treba zobrazit formulár
35         }
36     } else {
37         // tu treba zobrazit formulár
38     }
39
40     if ($zobraz_form) {
41         -?>
42         <form method="post">
43             <label for="meno">Meno:</label> <input type="text" name="meno"
                id="meno" value="<?php if (isset($_POST["meno"])) { echo
                $_POST["meno"]; } ?>"><br>
44             <label for="adresa">Adresa:</label> <textarea name="adresa" id
                ="adresa"><?php if (isset($_POST["adresa"])) { echo $_POST[
                "adresa"]; } ?></textarea><br>
45             <label for="vyber">vyber tovar:</label>
46             <select name="vyber" id="vyber">
47                 <option value="">-----</option>
48             <?php
49             if (isset($_POST["vyber"])) {
50                 vypis_tovarov($tovar_nazvy, $tovar_ceny, $_POST["vyber"]);
51             } else {
52                 vypis_tovarov($tovar_nazvy, $tovar_ceny, '');
53             }
54             -?>
55             </select><br>
56             <input type="submit" name="tlacidlo" value="Odošli údaje">
57         </form>
58     <?php
59     }
60     -?>

```

5.3 Kontrola vstupov od používateľa

Pri spracovaní formulárov je veľmi dôležité kontrolovať vstupy od používateľa (teda hodnoty, ktoré používateľ zadá do formulára, resp. môže zadať). Tejto problematike treba venovať zvýšenú pozornosť, lebo práve nedostatočnou kontrolou hodnôt vznikajú časté bezpečnostné problémy.

Ukážeme si dve rôzne kontroly:

1. či hodnoty spĺňajú nejaké pravidlá (správna dĺžka, číslo, korektná e-mailová adresa, ...) - kontrola textových polí
2. či hodnoty neobsahujú nežiadúce kódy (html značky, javascript, ...) - kontrola všetkých formulárových prvkov, aj tých, ktoré to na prvý pohľad nepotrebujú

Kontrola formátov vstupov

V textovom poli môžu používatelia zadávať rôzne údaje, väčšinou sú to textové informácie - meno, priezvisko, adresa, e-mailová adresa, ale môžu byť aj číselné, napr. dátum narodenia.



05/kontrola0
1.php

PRÍKLAD 5.17

Vytvoríme funkciu `spravne_meno($m)`, ktorá vráti `TRUE`, ak zadané meno bude mať dĺžku minimálne 3 znaky a maximálne 30 znakov. Inak vráti `FALSE`.

Na zistenie dĺžky reťazca použijeme reťazcovú funkciu `strlen()`.

Funkcia `strlen(reťazec)` vráti dĺžku reťazca alebo 0, ak je reťazec prázdny.

```
function spravne_meno($m) {  
    return (strlen($m) >= 3) && (strlen($m) <= 30);  
}
```

Upravíme aj kontrolu odosielania formulára. Už nebudeme kontrolovať, či `$_POST['meno']` `!= ''`, ale či má `$_POST` správny formát.

```
$zobraz_form = true;  
if (isset($_POST['tlacidlo'])) {  
    if (isset($_POST['meno']) && spravne_meno($_POST['meno'])) {  
        echo 'meno: ' . $_POST['meno'];  
        $zobraz_form = false;  
    } else {  
        echo 'Nezadal si meno alebo meno nemá správny formát!';  
    }  
}
```



05/kontrola0
2.php

PRÍKLAD 5.18

Vytvoríme funkciu `spravne_psc($psc)`, ktorá vráti `TRUE`, ak zadané PSČ bude mať presne 6 znakov a na 3. indexe je medzera. Inak vráti `FALSE`.

```
function spravne_psc($psc) {
    return (strlen($psc) == 6) && (strpos($psc, ' ') == 3);
}
```

PRÍKLAD 5.15

Vytvorme funkciu `spravny_vek($vek)`, ktorá vráti `TRUE`, ak zadaný vek je číslo väčšie ako 14. Inak vráti `FALSE`.



05/kontrola03.php

Parameter `$vek` môže byť reťazec s číslom, ak je údaj získaný z textového poľa formulára. Preto by sme potrebovali zistiť, či je daný parameter číslo alebo reťazec s číslom (teda, či sa dá daný reťazec korektne previesť na číslo). Na to slúži funkcia `is_numeric()`.

Funkcia `is_numeric($premenna)` zisťuje, či je zadaná premenná číslo alebo reťazec s číslom, teda vráti `TRUE` alebo `FALSE`.

Ešte potrebujeme pretypovať (prekonvertovať) reťazec na číslo, aby sme otestovali aj podmienku, že číslo je väčšie ako 14.

Ľubovoľnú premennú môžeme pretypovať na nami definovaný typ. Nemusí sa to vždy podariť. Napríklad, ak chceme prekonvertovať reťazec 'abc' na číslo, tak výsledkom bude 0. Pretypovanie robíme pomocou nasledujúceho zápisu: `(nový_typ) $premenna`.

Napr.:

<code>(int) '123'</code>	123
<code>(int) 13.4</code>	13
<code>(real) 13</code>	13.0

Naša funkcia by mohla vyzeráť nasledovne:

```
function spravny_vek($vek) {
    return is_numeric($vek) && ((int) $vek > 14);
}
```

DISKUSIA

Čo by sa mohlo ešte kontrolovať pri PŠČ? Aké podmienky by ste vymysleli na kontrolu telefónneho čísla?





ÚLOHA 5.20

Vytvorte funkciu `spravny_email($mail)`, ktorá vráti `TRUE`, ak je zadaný e-mail korektný. Inak vráti `FALSE`. E-mail je korektný, keď má minimálne 6 znakov a obsahuje znak `@` a niekde v reťazci za `@` je bodka.

Rovnaké kontroly môžeme použiť aj pre `<textarea>`.

Ošetrovanie vstupov

Ošetrovať používateľské vstupy môžeme viacerými spôsobmi – každý má svoje určenie.



PRÍKLAD 5.21

Zobrazme stránku z príkladu 5.17 a do textového poľa pre meno zadajme “`<h1>Janko</h1>`”. Čo sa zobrazí?

The screenshot shows a browser window titled 'Formulár' with the URL 'localhost/05/kontrola01.php'. The form contains a text input field labeled 'Meno:' with the value '<h1>Janko</h1>' and a button labeled 'Odošli údaje'.

Zobrazí sa:

The screenshot shows the rendered output of the form. The label 'meno:' is followed by a large, bold heading 'Janko'.

meno:

Janko

Janko sa nám zobrazil ako nadpis, čo asi nie je žiadúce, lebo to kazí vzhľad stránky. Ak by sme do textového poľa zadali len “`<h1>Janko`”, tak by nadpis ostal otvorený, a teda by sa takým štýlom mohli zobrazíť aj ďalšie informácie na stránke, ktoré by boli vypísané po rekapitulácii.

Potrebujeme teda zamedziť zobrazeniu HTML elementov. Na to slúži funkcia `htmlspecialchars(reťazec)`. Táto funkcia zmení preddefinované znaky na HTML entity. Preddefinované znaky sú: `&` (`&`), `"` (`"`), `'` (`'`), `<` (`<`), `>` (`>`). Napr.:

zadáme	bude v zdrojovom kóde
<code><h1>Janko</h1></code>	<code>&lt;h1&gt;Janko&lt;/h1&gt;</code>
<code><?php Janko ?></code>	<code>&lt;?php Janko ?&gt;</code>

To spôsobí, že zadané elementy stratia svoju funkciu a vypíšu sa len ako znaky. Táto funkcia sa využíva najmä vtedy, ak sa zadané hodnoty formulára vypisujú (hneď alebo neskôr) na stránke v rámci definovanej štruktúry.

PRÍKLAD 5.22

Upravme skript z príkladu 5.17 tak, aby sa neaplikovali prípadné HTML elementy zadané vo formulári.



05/kontrola04.php

```
if (isset($_POST['tlacidlo'])) {
    if (isset($_POST['meno']) && spravne_meno($_POST['meno'])) {
        echo 'meno: ' . htmlspecialchars($_POST['meno']);
        $zobraz_form = false;
    } else {
        echo 'Nezadal si meno alebo meno nemá správny formát!';
    }
}
```

Nekontrolovanie (neodfiltrovanie) HTML elementov by nemalo spôsobiť vážne problémy na stránke (a už vôbec nie na serveri), ale môže narušiť vzhľad a niekedy aj funkčnosť stránky. Toto sa často stáva pri rôznych diskusných fórach, ktoré si autori vytvárajú sami.

Ak chceme úplne odfiltrovať HTML (aj XML a PHP) elementy, použijeme funkciu `strip_tags(reťazec)`. Táto funkcia z reťazca odstráni všetky HTML elementy. Napr.:

zadáme	v zdrojovom kóde aj na webovej stránke
<code><h1>Janko</h1></code>	Janko
<code><?php Janko ?></code>	

V druhom prípade sa nezobrazí nič, lebo sa odstráni celá PHP vsuvka.

PRÍKLAD 5.23

Upravme predchádzajúci príklad (5.22) tak, aby sa odstránili prípadné html elementy zadané vo formulári.



05/kontrola05.php

```
if (isset($_POST['tlacidlo'])) {
    if (isset($_POST['meno']) && spravne_meno($_POST['meno'])) {
        echo 'meno: ' . strip_tags($_POST['meno']);
        $zobraz_form = false;
    } else {
        echo 'Nezadal si meno alebo meno nemá správny formát!';
    }
}
```



```
}
```

Ďalšia kontrola je už na inej úrovni. Používateľ totiž môže okrem HTML elementov zadať aj rôzne špeciálne znaky, ktoré síce nemajú v jazyku HTML žiadny význam, ale majú význam pri príkazoch vykonávaných na serveri. Neodfiltrovanie týchto špeciálnych znakov spolu so slabším zabezpečením servera, môže mať až katastrofálne následky. Niektoré špeciálne znaky môžeme eliminovať pomocou funkcie `addslashes(reťazec)`. Táto funkcia pridá (ako vyplýva z názvu) pred preddefinované znaky spätné lomítko `\`, čím znefunkční ich pôvodný význam a stanú sa z nich len bežné znaky. Preddefinované znaky sú: ' (apostrof), " (úvodzovka), \ (spätné lomítko) a NULL.



05/kontrola0
6.php

PRÍKLAD 5.24

Upravme príklad 5.22 tak, aby sa znefunkčnili špeciálne znaky.

```
if (isset($_POST['tlacidlo'])) {  
    if (isset($_POST['meno']) && spravne_meno($_POST['meno'])) {  
        echo 'meno: ' . addslashes($_POST['meno']);  
        $zobraz_form = false;  
    } else {  
        echo 'Nezadal si meno alebo meno nemá správny formát!';  
    }  
}
```

Silnejšou funkciou je `quotemeta(reťazec)`, ktorá pridáva spätné lomítko pred nasledujúce znaky: `. \ + * ? [^] ($)`.

Diskusia: Pri kontrole akých údajov by nám mohli tieto funkcie narobiť problémy?

Ako skĺbiť kontrolu hodnôt vstupov s ošetrovaním vstupov, teda, ako robiť kontrolu hodnôt na ošetrených vstupoch?



PRÍKLAD 5.25

Vytvoríme funkciu `osetri()`, ktorá ako parameter dostane reťazec a aj výsledkom bude reťazec, ktorý nebude obsahovať HTML elementy a pred špeciálnymi znakmi budú spätné lomítko.

```
function osetri($ret) {  
    return quotemeta(addslashes(trim(strip_tags($ret))));  
}
```

Na reťazec sme aplikovali predchádzajúce funkcie a okrem toho sme pridali aj funkciu `trim()`.

Funkcia `trim(reťazec)` zo začiatku a konca reťazca odstráni medzery, tabulátory, konce riadkov a ďalšie "biele znaky" (znaky, ktoré sa nezobrazujú).

PRÍKLAD 5.26

Dorobme kontrolu formulára a upravme aj formulár.



05/kontrola07.php

```
$zobraz_form = true;
if (isset($_POST['tlacidlo'])) {
    if (isset($_POST['meno'])) {
        $meno = osetri($_POST['meno']);
    } else {
        $meno = '';
    }
    if (spravne_meno($meno)) {
        echo "meno: $meno<br>\n";
        $zobraz_form = false;
    } else {
        echo "Nezadal si meno alebo meno nemá správny formát!<br>\n";
    }
}
```

Najskôr si do premennej `$meno` dáme ošetrený vstup (`osetri($_POST['meno'])`) alebo prázdny reťazec a potom overíme, či spĺňa požadované podmienky (`spravne_meno($meno)`).

Vo formulári už nebudeme kontrolovať `$_POST['meno']`, ale premennú `$meno`.

```
if ($zobraz_form) {
?>
<form method="post">
    <label for="meno">Meno:</label> <input type="text" name="meno"
id="meno" value="<?php if (isset($meno)) {echo $meno;} ?>"><br>
    <input type="submit" name="tlacidlo" value="Odošli údaje">
</form>
<?php
}
?>
```

Možno sa zdá, že položky, ktoré používateľ vyberá (tie, do ktorých nič nevypisuje, napr. výberová ponuka, začiarkávacie políčka, ...), netreba kontrolovať. To je však veľký omyl. Ak bude chcieť útočník napadnúť našu aplikáciu, vytvorí si kópiu nášho formulára, kde namiesto výberových prvkov vloží napr. textové polia (aby mohol zadávať príkazy). Útočník sa pokúsi odhadnúť, ako asi prebiehajú kontroly nášho formulára (prípadne ju vyskúša), zistí si povinné údaje a podľa toho si prispôsobí svoj formulár, ktorého spracovanie ale nasmeruje na náš formulár. Tým nám môže spôsobiť veľké škody.

PRÍKLAD 5.27

Majme formulár ako v príklade 5.15. Dorobme doňho overovanie hodnôt a kontroly. Položka meno (meno a priezvisko) musí mať minimálne 5 znakov, maximálne 50 znakov a musí obsahovať medzeru. Adresa musí mať minimálne 5 znakov, maximálne 100 znakov a tiež musí



05/kontrola08.php

obsahovať medzeru.

Okrem funkcie `osetri()` (z príkladu 5.25) pridáme funkciu `spravne_meno()` a `spravna_adresa()`.

```
function spravne_meno($m) {
    return (strlen($m) >= 5) && (strlen($m) <= 50) &&
        (strpos($m, ' ') > 0);
}

function spravna_adresa($m) {
    return (strlen($m) >= 5) && (strlen($m) <= 100) &&
        (strpos($m, ' ') > 0);
}
```



OTÁZKA

Vedeli by ste tieto dve funkcie nahradiť jednou?

Upravme overovanie formuláru.

```
if (isset($_POST['tlacidlo'])) {
    // ošetrenie vstupov a nastavenie premenných
    if (isset($_POST['meno'])) {
        $meno = osetri($_POST['meno']);
    } else {
        $meno = '';
    }
    if (isset($_POST['adresa'])) {
        $adresa = osetri($_POST['adresa']);
    } else {
        $adresa = '';
    }
    if (isset($_POST['vyber'])) {
        $vyber = osetri($_POST['vyber']);
    } else {
        $vyber = '';
    }
    if ((spravne_meno($meno) && (spravna_adresa($adresa)) &&
        ($vyber != '')) {
        // boli odoslané všetky údaje a zobrazíme rekapituláciu
        echo "meno: $meno<br>\n";
        echo "adresa: $adresa<br>\n";
        echo "tovar: {$tovar_nazvy[$vyber]}<br>\n";
        $zobraz_form = false; // tu formulár nezobrazíme
    } else {
        echo "Nezadal si niektorú položku alebo nemá správny
            formát!<br>\n";
    }
}
```

A na záver upravíme zobrazovanie nastavených hodnôt vo formulári.

```
<form method="post">
    <label for="meno">Meno:</label>
```

```

<input type="text" name="meno" id="meno" value="
<?php if (isset($meno)) {echo $meno;} ?>"><br>

<label for="adresa">Adresa:</label>
<textarea name="adresa" id="adresa" cols="30" rows="4">
<?php if (isset($adresa)) {echo $adresa;} ?>
</textarea><br>

<label for="vyber">vyber tovar:</label>
<select name="vyber" id="vyber">
<option value=''>-----</option>
<?php
if (isset($vyber)) {
    vypis_tovarov($tovar_nazvy, $tovar_ceny, $vyber);
} else {
    vypis_tovarov($tovar_nazvy, $tovar_ceny, '');
}
?>
</select><br>
<input type="submit" name="tlacidlo" value="Odošli údaje">
</form>

```

ÚLOHA 5.28



Vyrobte nasledujúci formulár.

Objednávka
×

Objednávka

Meno:

Adresa:

- Made Zigy (38.9 €)
- Sphero BB-8 Star Wars (100.9 €)
- LEGO Boost 17101 (141.9 €)
- OZOBOT 2.0 BIT (65.9 €)

osobný odber |
 kuriér |
 pošta

vytvoril: RoboHračky

Ošetríte v ňom všetky odosielané položky a skontrolujte meno, adresu (vymyslíte si vlastné kritériá) a skontrolujte, či bol zvolený aspoň 1 kus niektorého tovaru. Všetky položky formulára sú povinné.

Po korektnom odoslaní formulára sa zobrazí rekapitulácia:

Objednávka

Objednávka

Rekapitulácia objednávky:

meno: Janko Hraško
adresa: \? Halušková 10,\+ Hráškovo \\
Made Zigy: 1 ks
LEGO Boost 17101: 2 ks
odber: kurier

vytvoril: RoboHračky

Spätaná väzba pri odosielaní formulára



DISKUSIA

Aké správy by sa mali zobrazovať pri nekorrektne odoslanom formulári?

Ak chceme používateľom pomôcť pri vyplňaní formulárov, nestačí oznámiť, že formulár bol nesprávne vyplnený. Mali by sme ich upozorniť aj na chyby. Má to význam predovšetkým vtedy, ak formulár obsahuje viac položiek.



05/kontrola0
9.php

PRÍKLAD 5.29

Vychádzajme z riešenia príkladu 5.27 a doplníme zobrazovanie informácií o chybách vo formulári.

Najskôr vytvoríme funkciu, ktorá skontroluje, či je výber neprázdny.

```
function spravny_vyber($v) {  
    return !empty($v);  
}
```

Použili sme funkciu `empty($premenna)`, ktorá zisťuje, či je premenná prázdna. Premenná je prázdna, ak nie je definovaná alebo sa jej hodnota rovná `FALSE` (`""` - prázdny reťazec, `0` - `0` ako celé číslo, `0.0` - `0.0` ako reálne číslo, `"0"` - `0` ako reťazec, `[]` - prázdne pole, `FALSE`, `NULL`). Pripomeňme si, že výkričník pred funkciou znamená negáciu.

Zoznam chýb si budeme ukladať do asociatívneho poľa `$chyby`, ktorého kľúčmi budú názvy prvkov formulára (tých, v ktorých sa vyskytli chyby). Ak po skončení overovania formulára bude toto pole prázdne, tak sa vo formulári žiadne chyby nenachádzajú, formulár nezobrazíme a môžeme vypísať napr. rekapituláciu, inak vypíšeme chyby a znovu zobrazíme formulár.

Kontrolu existencie položky, jej ošetrovanie a kontrolu správnosti hodnoty sme spojili do jednej podmienky (spolu s else časťou), aby sme lepšie vedeli vypisovať chybu, kedy používateľ nezadal nič a kedy zadal hodnotu v nesprávnom formáte.

```
$zobraz_form = true;
$chyby = [];

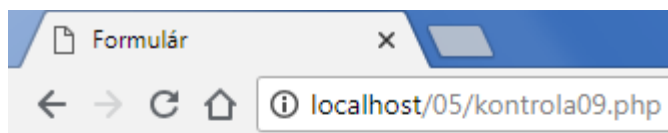
if (isset($_POST['tlacidlo'])) {
    if (isset($_POST['meno']) && $_POST['meno']) {
        $meno = osetri($_POST['meno']);
        if (!spravne_meno($meno)) {
            $chyby['meno'] = 'Meno nie je v správnom formáte.';
        }
    } else {
        $meno = '';
        $chyby['meno'] = 'Nezadal si meno.';
    }
}

if (isset($_POST['adresa']) && $_POST['adresa']) {
    $adresa = osetri($_POST['adresa']);
    if (!spravna_adresa($adresa)) {
        $chyby['adresa'] = 'Adresa nie je v správnom formáte.';
    }
} else {
    $adresa = '';
    $chyby['adresa'] = 'Nezadal si adresu.';
}

if (isset($_POST['vyber']) && $_POST['vyber']) {
    $vyber = osetri($_POST['vyber']);
    if (!spravny_vyber($vyber)) {
        $chyby['vyber'] = 'Nevybral si žiaden tovar.';
    }
} else {
    $vyber = '';
    $chyby['vyber'] = 'Nevybral si žiaden tovar.';
}

if (empty($chyby)) {
    echo "meno: $meno<br>\n";
    echo "adresa: $adresa<br>\n";
    echo "tovar: {$tovar_nazvy[$vyber]}<br>\n";
    $zobraz_form = false;
} else {
    echo "<b>Nekorektne vyplnený formulár!</b><br>\n";
    foreach($chyby as $hodnota) {
        echo "$hodnota<br>\n";
    }
}
}
```

Nesprávne vyplnený formulár by mohol vyzerať nasledovne:



Nekorektne vyplnený formulár!

Meno nie je v správnom formáte.

Adresa nie je v správnom formáte.

Nevybral si žiaden tovar.

Meno: Janko

Halušková10

Adresa:

vyber tovar: -----

Odošli údaje



ÚLOHA 5.30

Do úlohy 5.28 dorobte zobrazovanie chybových správ.

5.4 Metodické pokyny pre učiteľa

CIEĽ

Cieľom tejto časti je oboznámiť študentov s vylepšenou prácou s formulármi, ako je zobrazovanie už odoslaných hodnôt vo formulári a zobrazovanie/nezobrazovanie formulára. Okrem toho sa zoznámia so základnými funkciami, ktoré súvisia s bezpečnosťou pri kontrole formulárov.

VÝKLAD

Táto časť je zameraná na:

- zobrazovanie odoslaných údajov vo formulári,
- skrývanie (nezobrazovanie) formulára,
- základnú kontrolu a overovanie vstupov od používateľa.

Výklad sa strieda s riešenými príkladmi a samostatnými úlohami pre študentov.

Zobrazenie odoslaných údajov znovu vo formulári

K diskusii: Naviesť študentov na to, že používatelia očakávajú, že to, čo už raz vo formulári vyplnili, ostane vyplnené, aj keď formulár odošlú neúplný alebo nekorektne vyplnený.

K textarea: Treba si dávať pozor na medzery medzi značkami elementu textarea, lebo sa zachovávajú- Ak medzi ne vkladáme PHP kód tak medzi otváracou značkou elementu textarea a PHP kódom by nemala byť medzera, rovnako aj na konci, teda `<textarea><?php ... ?></textarea>`.

K výberovej ponuke: Treba si dávať pozor na to, že medzi atribútmi v `<option>` musí byť medzera. Teda buď bude medzera priamo v HTML kóde (za atribútom value a pred PHP vsuvkou), alebo v PHP vsuvke sa bude vypisovať text ' selected' (s medzerou na začiatku). To isté platí aj pre prepínače a začiarkávacie políčka.

Skrývanie formulára

K príkladu 5.15: So študentami diskutujte, prečo nie je dobré mať úplne rovnaký kód na rôznych miestach. Aké problémy to môže spôsobiť?

Kontrola vstupov od používateľa

Pri práci s formulármi platí zásadné pravidlo – **Nesmieme dôverovať vstupom od používateľa**. A preto treba každý prvok formulára testovať. Nesmieme sa totiž spoliehať na obmedzenia, ktoré sú vo formulári, napr. max. počet zadávaných znakov. Väčšina (nielen) začínajúcich tvorcov webových aplikácií (pri spracovaní formulárov) si myslí, že ak vo formulári môže používateľ zvoliť nejaké hodnoty, tak túto položku nemusí kontrolovať. To je však veľký omyl. Ak bude chcieť útočník napadnúť našu aplikáciu, vytvorí si kópiu nášho formulára, kde

namiesto výberových prvkov vloží napr. textové polia (aby mohol zadávať príkazy). Útočník sa pokúsi odhadnúť, ako asi prebiehajú kontroly nášho formulára (prípadne ju vyskúša) zistí si povinné údaje a podľa toho si prispôsobí svoj formulár, ktorý ale nasmeruje na náš formulár (atribút `action` v útočníkovom formulári bude obsahovať webovú adresu nášho formulára).

Funkcia `strip_tags(reťazec, povolene_elementy)` má aj voliteľný parameter `povolene_elementy`, v ktorom definujeme, ktoré elementy sa majú v reťazci zachovať.

Diskusia pri spätnej väzbe pri odosielaní formulára: Študentov treba naviesť na to, že nestačí správa o nekorektnom odoslaní formulára, ale aj informácia, pri ktorých položkách bola chyba. Môžete diskutovať aj o tom, kde a kedy by mali byť zobrazené informácie o požadovaných formátoch vyplňaných údajov.

6 PHP - VKLADANIE KÓDU Z RÔZNYCH SÚBOROV

Predstavme si situáciu, že máme niekoľko stránok, ktoré majú rovnaké časti, napríklad navigáciu alebo pätičku. V prípade statických HTML stránok musíme tieto časti vložiť do všetkých stránok. Ak však v niektorej časti chceme zmeniť text, musíme to urobiť na všetkých stránkach. Podobne je to napríklad pri spoločných premenných, konštantách alebo funkciách, ktoré chceme využívať na viacerých stránkach. V takýchto prípadoch si môžeme spoločné informácie uložiť do samostatného súboru a do konkrétnej stránky ich len vložiť (pripojiť) podobne ako napr. pri kaskádových štýloch.

6.1 Pripojenie súboru - INCLUDE

Na pripojenie (vloženie) obsahu súboru (prakticky ľubovoľného typu) k PHP súboru môžeme využiť funkciu `include(subor)`, kde parameter `subor` je kompletná (relatívna alebo absolútna) cesta k pripájanému súboru. Ten istý súbor nesmieme vložiť viackrát (priamo alebo z iného súboru), inak PHP interpret vyhlási chybu.

PRÍKLAD 6.1

Upravme stránky IT Pizza tak, aby hlavička, navigácia a päta boli v samostatných súboroch. Vychádzajme zo stránky `index.html` v priečinku `zdroj`.



06/zdroj/index.html

Aby sme však doň mohli "vložiť" obsah iného súboru, musíme ho premenovať na `index.php`.

Hlavičku spravíme ako PHP súbor, aby sme dynamicky mohli meniť titulok stránky. Súbor `hlavicka.php` bude obsahovať všetko od začiatku stránky po koniec elementu `<header>` zo súboru `index.php`:

06/riesenie1/hlavicka.php

```
<!doctype html>
<html>
<head>
...
</head>
<body>
  <header>
  ...
  </header>
```

Aj navigáciu spravíme ako PHP súbor, aby sme v nej mohli neskôr vypisovať aktuálny počet tovaru v nákupnom košíku. Súbor `navigacia.php` bude obsahovať celý element `<nav>`. V navigácii musíme zmeniť odkaz `index.html` na `index.php`:

06/riesenie1/navigacia.php

```
<nav>
  <div id="kosik_nav">
  ...
  </div>
  <a href="index.php">ponuka</a>
  <a href="galeria.html">fotogaléria</a>
  <a href="rezervacia.html">rezervácia</a>
  <a href="praca.html">práca</a>
```

```
</nav>
```

Súbor `pata.html` bude obsahovať všetko od elementu `<footer>` po koniec súboru:

```
<footer id="kontakt">
  <h2>Kontakt</h2>
  ...
</footer>
</body>
</html>
```

Teraz môžeme upraviť súbor `index.php`. Zmažeme z neho celú časť, ktorá reprezentuje hlavičku, navigáciu a päť a namiesto nich vložíme súbory `hlavicka.php`, `navigacia.php` a `pata.html`:

```
<?php
include('hlavicka.php');
include('navigacia.php');
?>
<section id="ponuka">
  <h2>Ponuka</h2>
  ...
</section>
<aside id="akcia">
  ...
</aside>
<?php
include('pata.html');
?>
```

Keď teraz zobrazíme `index.php`, stránka by mala vyzerať rovnako ako HTML verzia, z ktorej sme vychádzali.

Keď sa pozrieme na zdrojový kód stránky, tak budeme vidieť, že obsahuje všetky informácie z vkladaných súborov a aj zo súboru `index.php`.

To, čo sme spravili v súbore `index.php`, je akási šablóna:

```
<?php
include('hlavicka.php');
include('navigacia.php');
?>
<!-- obsah hlavnej časti -->
<?php
include('pata.html');
?>
```

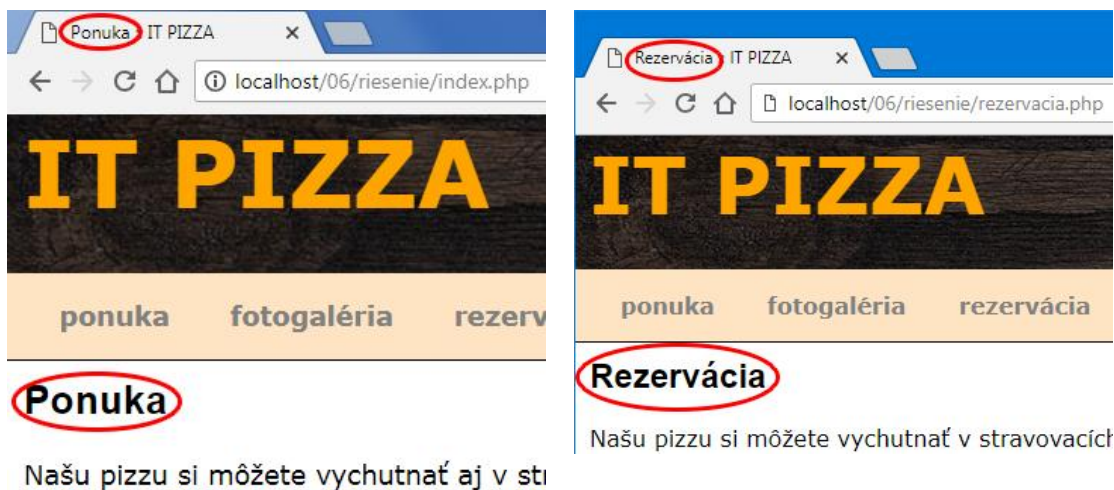
Túto šablónu môžeme použiť na každú stránku webového sídla IT pizza. Meniť sa bude len obsah hlavnej časti.



ÚLOHA 6.2

Upravte stránku `galeria.html` tak, aby ste na hlavičku, navigáciu a päť použili súbory ako v príklade 6.1. Nezapudnite upraviť aj súbor `navigacia.php`.

Chceli by sme, aby sa titulok stránky menil podľa aktuálnej stránky. Okrem toho chceme zabezpečiť, aby sa rovnaký nadpis zobrazoval aj v obsahu hlavnej stránky, napr.:



PRÍKLAD 6.3

Upravme stránku ponuky tak, aby sa zobrazoval rovnaký text v titulku stránky aj v hlavnom nadpise obsahu stránky.

Musíme upraviť súbory `index.php` a aj `hlavicka.php`.

Na začiatok stránky `index.php` pridáme premennú `$nadpis`, ktorú potom vypíšeme v elemente `<h2></h2>`.

```
<?php
$nadpis = 'Ponuka';
include('hlavicka.php');
include('navigacia.html');
?>
<section id="ponuka">
  <h2><?php echo $nadpis; ?></h2>
```

V súbore `hlavicka.php` doplníme do titulku vypisovanie premennej `$nadpis` (ktorú sme definovali v súbore `index.php` a definovali sme ju ešte pred vložením súboru `hlavicka.php`).

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title><?php echo $nadpis; ?> :: IT PIZZA</title>
  <link href="styly.css" rel="stylesheet">
```

Prečo to bude fungovať, keď premennú máme definovanú len v súbore `index.php`?

Súbor `hlavicka.php` nespúšťame v prehliadači samostatne, ale je súčasťou `index.php` (lebo sme ho do `index.php` vložili). Vykonáva sa len `index.php`, ktorý má prístup k premennej aj k obsahu hlavičky.



06/riesenie2/index.php

06/riesenie2/hlavicka.php



ÚLOHA 6.4

Do stránky `galeria.php` dorobte správne zobrazovanie titulku a nadpisu v hlavnej obsahovej časti.

6.2 Oddelenie obsahu stránky od funkcií

Väčšinou to budú údaje získané z databázy. My ich však budeme mať uložené v poliach.



PRÍKLAD 6.5

Vytvoríme súbor s údajmi o pizziach a pripojíme ho tak, aby sme ho mohli využiť na stránke `index.php`. Potom na stránke `index.php` vygenerujeme jednotlivé elementy `<article>`, ktoré reprezentujú pizze.

06/riesenie3/udaje.php

V súbore `udaje.php` údaje o pizziach rozdelíme do piatich polí.

```

<?php
$nazvy = ['Margherita', 'Cardinale', 'Funghi', 'Hawai',
'Prosciutto', 'Quattro Formaggi', 'Tonno'];
$popisy = ['paradajková omáčka, syr', 'paradajková omáčka, syr,
šunka', 'paradajková omáčka, syr, šampiňony', 'paradajková omáčka,
syr, šunka, ananás', 'paradajková omáčka, mozarella, šunka',
'paradajková omáčka, 4 druhy syra', 'paradajková omáčka, mozarella,
tuniak, cibuľa'];
$ ceny = [3.15, 3.99, 5.05, 4.10, 5.89, 4.50, 4.05];
$alergeny = ['1, 7', '1, 7', '1, 7', '1, 7', '1, 7', '1, 7', '1, 4,
7'];
$subory = ['margerita.jpg', 'cardinale.jpg', 'funghi.jpg',
'hawai.jpg', 'prosciutto.png', 'formaggi.png', 'tonno.png'];
?>

```

Rovnaký index v každom poli reprezentuje údaje o jednej pizzi.

06/riesenie3/hlavicka.php

Súbor `udaje.php` vložíme na začiatok súboru `hlavicka.php`. Tým zabezpečíme, že ho nemusíme vkladať do ďalších súborov, v ktorých by sme tieto údaje mohli potrebovať. Budú prístupné vo všetkých súboroch, ktoré majú vložený súbor `hlavicka.php`.

```

<?php
include('udaje.php');
?>
<!doctype html>
<html>

```

06/riesenie3/index.php

Na stránke `index.php` môžeme teraz vygenerovať informácie o jednotlivých pizziach.

```

<?php
foreach($nazvy as $kluc => $hodnota) {
    echo "<article>\n";
}

```

```

    echo "<h3>${hodnota} ({$ceny[$kluc]} &euro;)</h3>{$popisy[$kluc]}
<sub>{$alergeny[$kluc]}</sub><br>\n";
    echo '<form method="post">';
    echo '<button type="submit" name="pizza" value="" . $kluc .
'">' . "</button>\n";
    echo "</form>\n";
    echo '<br>';
    echo "</article>\n";
}
?>

```

Rovnako, ako sme údaje vyčlenili do samostatného súboru, môžeme tak spraviť aj s funkciami. Mnohé funkcie vieme využiť na viacerých stránkach, a tiež naše súbory budú prehľadnejšie, keď nebudú obsahovať funkcie.

PRÍKLAD 6.6

Upravme stránku `rezervacia.html` tak, aby sa hlavička, navigácia a päta vkladala z príslušných súborov. Potom upravme formulár - jeho spracovanie aj overovanie. Funkcie, ktoré budeme potrebovať, uložíme do súboru `funkcie.php`, ktorý, rovnako ako `udaje.php`, vložíme do hlavičky.

Najskôr premenujme súbor na `rezervacia.php` a odstráňme spoločné informácie pre všetky stránky. Pridajme aj premennú `$nadpis`.

```

<?php
$nadpis = 'Rezervácia';
include('hlavicka.php');
include('navigacia.php');
?>
<section id="rezervacia">
  <h2><?php echo $nadpis; ?></h2>
  <div>
    ...
  </div>
  <form method="post">
    ...
  </form>
</section>
<?php
include('pata.html');
?>

```

Nesmieme zabudnúť upraviť aj stránku `navigacia.php`.

```

<nav>
  ...
  <a href="rezervacia.php">rezervácia</a>
  <a href="praca.html">práca</a>
</nav>

```

Vráťme sa k súboru `rezervacia.php`. Upravme elementy `<select>` pre počet osôb, deň, rok a hodiny tak, aby jednotlivé elementy `<option>` vygenerovala funkcia



06/riesenie4/navigacia.php

06/riesenie4/funkcie.php

`vypis_poloziiek()` z príkladu 5.6. Vytvoríme súbor `funkcie.php`, skopírujme doň funkciu `vypis_poloziiek()` a pridáme nastavenie časovej zóny.

```
<?php
date_default_timezone_set('Europe/Bratislava');
function vypis_poloziiek($od, $do, $oznac) {
    ...
}
?>
```

06/riesenie4/hlavicka.php

Súbor `funkcie.php` pripojme do `hlavicka.php`.

```
<?php
include('udaje.php');
include('funkcie.php');
?>
<!doctype html>
<html>
    ...
</html>
```

06/riesenie4/revervacia.php
06/riesenie4/...

Teraz upravme súbor `rezervacia.php`. Nahradíme vypisovanie elementov `<option>` funkciou. Počet osôb bude mať hodnoty od 0 do 10, deň od 1 do 31, rok aktuálny rok a nasledujúci rok a hodiny budú od 10 do 20. Pre počet osôb nebude prednastavená žiadna hodnota (teda 0), pre ostatné aktuálny deň, rok a hodina.

```
<form method="post">
    ...
    <select name='pocet'>
        <?php echo vypis_poloziiek(0, 10, 0); ?>
    </select><br>
    ...
    Dátum: <select name='den'>
        <?php echo vypis_poloziiek(1, 31, date("j")); ?>
    </select>
    <select name='mesiac'>
        ...
    </select>
    <select name='rok'>
        <?php
        $rok = date("Y");
        echo vypis_poloziiek($rok, $rok + 1, $rok);
        ?>
    </select><br>
    Čas: <select name='hod'>
        <?php echo vypis_poloziiek(10, 20, date("G")); ?>
    </select>
    ...
</form>
```



ÚLOHA 6.7

Upravte vypisovanie mesiacov (z príkladu 6.6) tak, aby bol vo formulári prednastavený aktuálny mesiac. Môžete si na výpis mesiacov vytvoriť funkciu napr. `vypis_mesiacov($oznac)`, podobne ako je funkcia `vypis_poloziiek()`. Na začiatku funkcie vytvorte pole, ktoré bude

obsahovať názvy mesiacov. Hodnotami môžu byť čísla.

POZNÁMKA

Užitočná môže byť aj funkcia `vypis($hodnoty, $texty, $oznac)`, kde `$hodnoty` a `$texty` sú polia. Prípadne `vypis1($texty, $oznac)`, ak by nám stačili číselné hodnoty v elementoch `<option>`. Touto funkciou by sme mohli nahradiť funkciu `vypis_mesiacov()`, kde pole názvov mesiacov by sme mohli doplniť do súboru `udaje.php` a ako parameter ho zadať tejto funkcii.



PRÍKLAD 6.8

Do súboru `rezervacia.php` doplníme ešte odosielanie a overovanie formulára. Overovať budeme len položky `meno`, `priezvisko` a `telefon` a či používateľ zadal nenulový počet osôb a vybral mesto.



Do súboru `funkcie.php` skopírujme funkcie `osetri()` a `spravne_meno()` z príkladu 5.29. Z funkcie `spravne_meno()` odstránime poslednú podmienku. Ďalej dopíšme funkciu `spravny_telefon($t)`. V tejto funkcii skontrolujeme, či premenná začína reťazcom `+42` a či má dĺžku aspoň 13 znakov.

06/riesenie5/funkcie.php

```
function osetri($ret) {
    return quotemeta(addslashes(trim(strip_tags($ret))));
}
function spravne_meno($m) {
    return (strlen($m) >= 5) && (strlen($m) <= 50);
}
function spravny_telefon($t) {
    return (strlen($t) >= 13) && (strpos($t, '+42') === 0);
}
```

Keďže chceme, aby telefón začínal presne znakmi `+42` použijeme porovnanie `===`. Ak by sme použili porovnanie `==`, tak by stačilo zadať dostatočne dlhý reťazec a funkcia by dala hodnotu `TRUE`. Pri neexistencii tohto podreťazca by funkcia vrátila `FALSE`, ale vďaka automatickej konverzii pri porovnaní by porovnanie s hodnotou `0` bolo `TRUE`, čo nechceme.

V súbore `rezervacia.php` doplníme odosielanie a overovanie formulára.

```
<div>
...
<?php
$zobraz_form = true;
$chyby = [];
if (isset($_POST['tlacidlo'])) {
    // kontrola mena
    if (isset($_POST['meno']) && $_POST['meno']) {
        $meno = osetri($_POST['meno']);
        if (!spravne_meno($meno)) {
```



```

        $chyby['meno'] = 'Meno nie je v správnom formáte.';
    }
} else {
    $meno = '';
    $chyby['meno'] = 'Nezadal si meno.';
}
// kontrola priezviska
if (isset($_POST['priezvisko']) && $_POST['priezvisko']) {
    $priezvisko = osetri($_POST['priezvisko']);
    if (!spravne_meno($priezvisko)) {
        $chyby['priezvisko'] = 'Priezvisko nie je v správnom
formáte.';
    }
} else {
    $priezvisko = '';
    $chyby['priezvisko'] = 'Nezadal si priezvisko.';
}
// kontrola telefónu
if (isset($_POST['tel']) && $_POST['tel']) {
    $tel = osetri($_POST['tel']);
    if (!spravny_telefon($tel)) {
        $chyby['tel'] = 'Telefón nie je v správnom formáte.';
    }
} else {
    $tel = '';
    $chyby['tel'] = 'Nezadal si telefón.';
}
// kontrola počtu
if (!$_POST['pocet']) {
    $chyby['pocet'] = 'Nevybral si počet osôb.';
}
// kontrola mesta
if (!isset($_POST['mesto'])) {
    $chyby['mesto'] = 'Nevybral si mesto.';
}
// ak nie sú žiadne chyby, zobrazíme rekapituláciu
if (empty($chyby)) {
    echo "meno: $meno<br>\n";
    echo "priezvisko: $priezvisko<br>\n";
    echo "telefón: $tel<br>\n";
    echo "počet osôb: {$_POST['pocet']}<br>\n";
    echo "mesto: ";
    if ($_POST['mesto'] == 'ba') {
        echo "Bratislava<br>\n";
    } else {
        echo "Banská Bystrica<br>\n";
    }
}
$zobraz_form = false; // tu formulár nezobrazíme
} else {
    ...
}
}
?>
</div>
<?php
if ($zobraz_form) {
?>
<form method="post">
    ...
    <select name='pocet'>
        <?php
        if (isset($_POST['pocet'])) {
            echo vypis_poloziek(0, 10, $_POST['pocet']);
        } else {

```

```

        echo vypis_poloziiek(0, 10, 0);
    }
    ?>
</select><br>
...
Dátum: <select name='den'>
    <?php
    if (isset($_POST['den'])) {
        echo vypis_poloziiek(1, 31, $_POST['den']);
    } else {
        echo vypis_poloziiek(1, 31, date("j"));
    }
    ?>
</select>
...
<select name='rok'>
    <?php
    $rok = date("Y");
    if (isset($_POST['rok'])) {
        echo vypis_poloziiek($rok, $rok + 1, $_POST['rok']);
    } else {
        echo vypis_poloziiek($rok, $rok + 1, $rok);
    }
    ?>
</select><br>
Čas: <select name='hod'>
    <?php
    if (isset($_POST['hod'])) {
        echo vypis_poloziiek(10, 20, $_POST['hod']);
    } else {
        echo vypis_poloziiek(10, 20, date("G"));
    }
    ?>
</select>
...
</form>
<?php
}
?>

```

Aj napriek tomu, že formulár odošleme so správnym telefónnym číslom, vypíše sa chybová správa, že telefón nie je v správnom formáte. Problémom je znak **+** na začiatku telefónu, pred ktoré dá funkcia `quotemeta()` spätné lomítko. Tento problém vyriešime tak, že do premennej `$tel` dáme ošetrovaný reťazec bez prvého znaku (teda v prípade správne zadaného telefónu funkcia `quotemeta()` pridá spätné lomítko `\`, ktoré odstránime).

```

if (isset($_POST['tel']) && $_POST['tel']) {
    $tel = substr(osestri($_POST['tel']), 1);
    if (!spravny_telefon($tel)) {
        $chyby['tel'] = 'Telefón nie je v správnom formáte.';
    }
} else {
    $tel = '';
    $chyby['tel'] = 'Nezadal si telefón.';
}

```



ÚLOHA 6.9

Do predchádzajúceho príkladu doplňte zobrazovanie všetkých vyplnených údajov vo formulári a v rekapitulácii vypisovanie ostatných údajov.



ÚLOHA 6.10

Doplňte stránku `praca.html` o funkčnosť formulára aj s jeho ošetrovaním a vypisovaním chýb pri nekorektne zadaných údajoch. Nezabudnite na správne zobrazovanie titulku a nadpisu stránky.



ÚLOHA 6.11

Upravte ostatné stránky IT pizza tak, aby sa na spoločné údaje použilo vkladanie príslušných súborov.

6.3 Metodické pokyny pre učiteľa

CIEĽ

Cieľom tejto časti je oboznámiť študentov s možnosťami vkladania rôznych súborov do PHP kódu. Dokážeme tak sprehľadniť a zefektívniť kód. Zameriame sa predovšetkým na vkladanie PHP kódu prípadne HTML kódu.



VÝKLAD

Táto časť je zameraná na:

- vkladanie iného kódu (HTML alebo PHP) z externých súborov do PHP kódu.

Výklad sa strieda s riešenými príkladmi a samostatnými úlohami pre študentov.



[K príkladu 6.1:](#)

Pri všetkých súboroch, ktoré vkladme do PHP súborov treba kontrolovať kódovanie, aby bolo rovnaké ako je PHP súbor, do ktorého vkladáme (UTF 8).

Hlavičku sme uložili do súboru `hlavicka.php`, pričom by súbor mohol zostať aj `hlavicka.html`. Pre študentov by to však mohlo byť mätúce, že v HTML súbore je PHP vsuvka. Na prvý pohľad by to nemalo fungovať. Ale keďže súbor `hlavicka.php` (resp. `hlavicka.html`) nevoláme priamo, ale vkladáme do `index.php`, tak PHP kód sa vykonáva až v súbore `index.php`.

[K príkladu 6.8:](#)

Odstránenie spätných lomítok z funkcie `quotemeta()` môžeme aj pomocou funkcie `stripslashes()`.

7 PHP - TRVALÉ UKLADANIE ÚDAJOV

Doteraz sme si ukázali ako spracovať formulár, ale zatiaľ sme odoslané údaje z neho nikde neukladali. To si ukážeme v tejto kapitole. Možností, ako trvale ukladať údaje, je niekoľko - do súborov, do databáz,... Dnešným štandardom je ukladanie údajov do databáz, ale problematika PHP a databázy je príliš rozsiahla, preto sa jej v tejto učebnici nebudeme venovať.

So súbormi budeme pracovať podobne ako s textovými súbormi v nejakom inom programovacom jazyku. Do súboru teda môžeme zapisovať len čísla a reťazce. Ak chceme do súboru zapísať inú štruktúru, musíme ju prekonvertovať na tieto typy.

Pri zapisovaní do súborov si musíme stanoviť, v akom formáte si budeme jednotlivé údaje v súbore pamätať. Predpokladajme, že máme objednávkový formulár s údajmi meno, adresa a tovar. Máme niekoľko možností:

- pamätať si každý údaj v samostatnom riadku, napr.
Janko Hraško
Cícerová 27 940 01 Košice
Lego Mindstorms,
- pamätať si všetky údaje z jednej objednávky v jednom riadku, oddelené konkrétnym oddeľovačom, napr.
Janko Hraško#Cícerová 27 940 01 Košice#Lego Mindstorms
- pamätať si údaje v nejakej štruktúre, napr. XML, JSON, ...,
- ...

Každý zo spôsobov má svoje výhody aj nevýhody.

DISKUTUJTE

Diskutujte o výhodách a nevýhodách rôznych spôsobov uchovávaní údajov.



7.1 Uloženie odoslaných údajov do súboru

PRÍKLAD 7.1

Budeme dopĺňať kód, ktorý sme vytvorili v rámci príkladu 5.29. Po odoslaní formulára zapíšeme *meno* do súboru `objednavka.txt`.



07/objednavka-zapis01.php

Zapísanie údajov urobíme v časti `if (empty($chyby)) { }`, kde sme vypísali rekapituláciu objednávky. Tá zostane, ale pridáme zapísanie do súboru.

Na zapísanie údajov do súboru môžeme využiť funkciu `file_put_contents(súbor, údaje)`. Táto funkcia zapíše reťazec alebo pole do súboru. Ak súbor neexistuje, vytvorí sa. Ak už súbor existoval, vymaže sa všetok obsah a nahradí sa novým (prepíše sa). Súbor zadávame ako názov súboru, prípadne spolu s cestou.

Neodporúčame zapisovať pole, lebo všetky prvky sa zapíšu do jedného riadka bez akéhokoľvek oddeľovača.

```

if (empty($chyby)) {
    echo "meno: $meno<br>\n";
    echo "adresa: $adresa<br>\n";
    echo "tovar: {$tovar_nazvy[$vyber]}<br>\n";
    $zobraz_form = false; // tu formulár nezobrazíme
    // zapíš údaje do súboru
    file_put_contents('objednavka.txt', $meno);
} else {
    ...
}

```

Súbor `objednavka.txt` bude vyzeráť nasledovne:

```

objednavka-zapis01.php  objednavka.txt
1 Janko Hraško

```



07/objednavka-zapis02.php

PRÍKLAD 7.2

Zapíšme do súboru `objednavka.txt` všetky údaje z objednávky (*meno, adresa, tovar*). Každý údaj zapíšme na samostatný riadok.

Musíme si uvedomiť, že pomocou funkcie `file_put_contents()` nemôžeme zapísať každý údaj samostatne, lebo funkcia súbor najprv vymaže. Teda by sa nám zachoval len posledný údaj. Preto si najprv pripravíme reťazec, v ktorom sú jednotlivé údaje oddelené znakmi konca riadku a až ten nakoniec zapíšeme do súboru.

```

if (empty($chyby)) {
    ...
    // zapíš údaje do súboru
    $udaje = $meno . "\n" . $adresa . "\n" . $vyber;
    file_put_contents('objednavka.txt', $udaje);
} else {
    ...
}

```

Súbor `objednavka.txt` bude vyzeráť nasledovne:

```

objednavka-zapis02.php  objednavka.txt
1 Janko Hraško
2 Halušková 10, Hraškovo
3 MZ

```



ÚLOHA 7.3

Urobte niekoľko objednávok z aplikácie v príklade 7.2 a po každej si pozrite súbor `objednavky.txt`. Skúšajte rôzne dlhé texty v položke *adresa*. Platí pri všetkých pravidlo, že jeden riadok obsahuje informácie o jednej položke?

Ak budeme do položky *adresa* vkladať len údaje v jednom riadku, bude všetko v poriadku. Ak však do adresy vložíme niekoľko skutočných riadkov (stlačíme kláves *Enter*), potom vo výstupnom súbore budeme mať viac ako tri riadky (adresa bude zaberáť viac riadkov). Toto však môže spôsobiť problém pri čítaní súboru, lebo nebudeme vedieť, koľko riadkov máme prečítať. Riešenií je niekoľko. Jedným z nich je, že pred zápisom adresy do súboru konce riadkov odstránime, resp. ich nahradíme iným znakom (napr. čiarkou alebo bodkočiarkou).

My si ukážeme iný spôsob, ktorý zachová pôvodné formátovanie údajov. Riešením je využiť štruktúru **JSON** (JavaScript Object Notation), ktorá je dnes bežná v rôznych webových aplikáciách a vieme s ňou pracovať v rôznych programovacích jazykoch.

7.2 JSON

JSON (JavaScript Object Notation) je jeden z najpoužívanejších formátov pre výmenu údajov (aj rôznych typov) na webe. Ľudia ho dokážu ľahko čítať a aj doň zapisovať. Programy ho dokážu ľahko analyzovať aj generovať.

Má veľmi dobrú podporu v programovacích jazykoch, napr. C, C++, C#, Delphi, Java, JavaScript, Lisp, Matlab, Perl, PHP (od 5.2 natívne), Python, Ruby...

Do JSON môžeme uložiť tieto typy: číslo (celé, reálne), textový reťazec v úvodzovkách, logickú hodnotu, pole, objekt. Príklady:

- [0, "PHP", false] - jednoduché pole
- ['MZ':38.90, 'SSW':100.90, 'LB': 141.90, 'OB':65.90] - asociatívne pole
- {"osoba": {"name": "Web", "age": 25}, "login": false, "opravenia": [1,2,8]} - objekt

7.2.1 Využitie JSON v PHP

V PHP máme funkciu `json_encode()`, ktorá dokáže pretransformovať ľubovoľnú premennú do formátu JSON - vytvorí jeden reťazec.

Všetky tri údaje z objednávky (meno, adresa, tovar) chceme zapísať do jedného riadku. Preto z nich najskôr vytvoríme pole, ktoré potom prevedieme do JSON formátu a zapíšeme do súboru.

PRÍKLAD 7.4

Zapíšme do súboru `objednavka.txt` všetky údaje z objednávky (*meno, adresa, tovar*) vo formáte JSON.



07/objednavka-zapis03.php

Všetky údaje si najprv vložíme do poľa `$udaje`. To následne pretransformujeme do formátu JSON pomocou funkcie `json_encode($udaje)` a výsledný reťazec zapíšeme do súboru.

```
if (empty($chyby)) {  
    ...  
    // zapíš údaje do súboru  
    $udaje = ['meno' => $meno, 'adresa' => $adresa, 'vyber' =>  
    $vyber];  
    $udaje_json = json_encode($udaje);  
    file_put_contents('objednavka.txt', $udaje_json);  
} else {
```



```
...
```

Súbor `objednavka.txt` bude vyzeráť nasledovne:

```
objednavka-zapis03.php  objednavka.txt
1 {"meno":"Janko Hrasko","adresa":"Haluskova 10,\r\nHraskovo","vyber":"MZ"}
```

Ak píšeme aj s diakritikou, tak súbor bude vyzeráť:

```
objednavka-zapis03.php  objednavka.txt
1 {"meno":"Janko Hra\u0161ko","adresa":"Halu\u0161kov\u00e1 10,\r\nHr\u00e1\u0161kovo","vyber":"MZ"}
```

Všimnite si, že diakritické znamienka sú kódované.

7.3 Načítanie údajov zo súboru

Ukázali sme si, ako údaje zapisovať do súboru. Poďme ich teraz načítať a zobraziť na stránke.



07/objednavka-vypis01.php

PRÍKLAD 7.5

Vypíšeme objednávku, ktorá je v súbore `objednavka.txt` uložená vo formáte JSON. Z pôvodnej objednávky potrebujeme len polia s názvami a cenami tovaru.

Obsah súboru načítame pomocou funkcie `file_get_contents(súbor)`, ktorá načíta celý obsah súboru ako jeden reťazec. Keďže v súbore je len JSON reťazec, tak ho dekodujeme pomocou funkcie `json_decode(reťazec)`. Táto funkcia reťazec štandardne konvertuje na objekt, čo nám nevyhovuje. Avšak funkcii `json_decode()` môžeme dať druhý parameter (typu *Boolean*) `json_decode(reťazec, TRUE / FALSE)`, ktorý keď nastavíme na `TRUE`, tak funkcia dekoduje reťazec ako asociatívne pole. S takým poľom už môžeme ďalej pracovať. Mali by sme dostať rovnaké pole, ktoré sme si pripravili pred zápisom do súboru. Zatiaľ ho vypíšeme pomocou funkcie `print_r()`.

```
<body>
<?php
// vypíš údaje zo súboru
$udaje_json = file_get_contents('objednavka.txt');
$udaje = json_decode($udaje_json, true);
print_r($udaje);
?>
</body>
```

Vidíme, že funkcia pole vypíše (dostala korektné pole) a obsahuje 3 kľúče - *meno*, *adresa*, *vyber*.

```
Výpis zo súboru x
localhost/07/objednavka-vypis01.php
```

```
Array ( [meno] => Janko Hraško [adresa] => Halušková 10, Hraškov [vyber] => MZ )
```

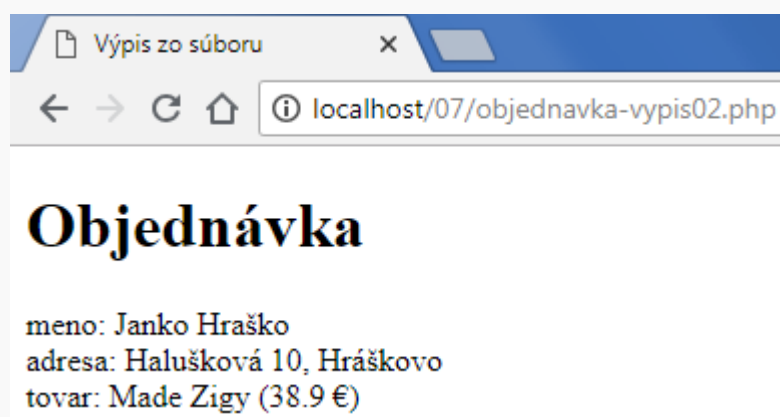
Čo by sa však stalo, keby súbor neexistoval? Funkcia `file_get_contents()` by vypísala upozornenie (*Warning*). Preto by sme nemali zo súboru čítať, pokým si nie sme istí, že existuje.

Na otestovanie, či súbor existuje, môžeme využiť funkciu `file_exists(súbor)`, ktorá vráti `TRUE`, ak súbor existuje, resp. `FALSE`, ak súbor neexistuje.

```
<?php
// test, či súbor existuje
if (file_exists('objednavka.txt')) {
    // vypíš údaje zo súboru
    $udaje_json = file_get_contents('objednavka.txt');
    $udaje = json_decode($udaje_json, true);
    print_r($udaje);
} else {
    echo 'Vstupný súbor neexistuje.';
}
?>
```

PRÍKLAD 7.6

Upravme výpis objednávky, aby vyzeral ako na nasledujúcom obrázku.



07/objednavka-vypis02.php

Meno a adresu vypíšeme štandardne cez premenné `$udaje['meno']` a `$udaje['adresa']`. Názov tovaru a jeho cenu však musíme zistiť podľa kľúča z polí `$tovar_nazvy` a `$tovar_ceny`. Kľúčom bude hodnota `$udaje['vyber']`.

```
echo "<h1>Objednávka</h1>\n";
echo "meno: {" . $udaje['meno'] . "<br>\n";
echo "adresa: {" . $udaje['adresa'] . "<br>\n";
echo "tovar: {" . $tovar_nazvy[$udaje['vyber']] .
    (" {" . $tovar_ceny[$udaje['vyber']] . " €)<br>\n";
```

Objednávka sa nám pekne vypíše. Formátovať ju môžeme ľubovoľne pomocou HTML elementov.

Avšak adresa je v jednom riadku. Pričom, keď sme ju odosielali a ukladali do súboru, bola v dvoch riadkoch. Využitím JSON formátu sme chceli toto riadkovanie zachovať, čo sa nám zatiaľ nepodarilo. Ak si však pozrieme zdrojový kód výpisu objednávky, zistíme, že v zdrojovom kóde je *adresa* v dvoch riadkoch.

```

8 <h1>Objednávka</h1>
9 meno: Janko Hraško<br>
10 adresa: Halušková 10,
11 Hráškovo<br>
12 tovar: Made Zigy (38.9 €)<br>

```

Teda zalomenie riadka (`\n`) sa zachovalo, ale aby sme to videli aj v prehliadači, musíme využiť štandardnú PHP funkciu `n12br (reťazec)`, ktorá v zadanom reťazci nahradí všetky znaky `\n` reťazcami `
`. Takto upravený reťazec bude výsledkom funkcie `n12br ()`.



07/objednavka-vypis03.php

PRÍKLAD 7.7

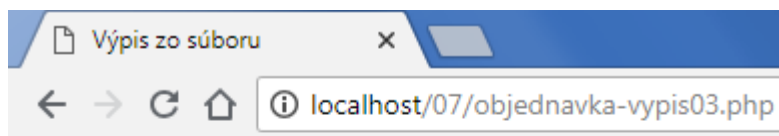
Upravme výpis adresy objednávky pomocou funkcie `n12br ()`.

```

echo "<h1>Objednávka</h1>\n";
echo "meno: {$udaje['meno']}<br>\n";
echo "<strong>adresa:</strong> " . n12br ($udaje ['adresa']) .
"<br>\n";
echo "tovar: {$tovar_nazvy[$udaje['vyber']]
({$tovar_ceny[$udaje['vyber']] } €)<br>\n";

```

Výpis môže vyzerat nasledovne:



Objednávka

meno: Janko Hraško
adresa: Halušková 10,
 Hráškovo
tovar: Made Zigy (38.9 €)



ÚLOHA 7.8

Do formulára z úlohy 5.28 dorobte zapisovanie do súboru a vytvorte aj stránku, ktorá zobrazí zapísané údaje..

7.4 Metodické pokyny pre učiteľa

CIEĽ

Cieľom tejto časti je oboznámiť študentov s jednoduchou prácou so súbormi (čítanie a zapisovanie). Takisto sa oboznámia s formátom JSON.

VÝKLAD

Táto časť je zameraná na:

- zápis do súboru
- čítanie zo súboru
- kódovanie údajov do JSON formátu a jeho spätné dekódovanie do PHP premenných

Výklad sa strieda s riešenými príkladmi a samostatnými úlohami pre študentov.

Problematike samotných databáz sa venuje predmet *Databázy*. Uploadom súborov sa v tejto učebnici nezaobráame.

Všetky príklady v kapitole zapisujú len 1 objednávku do súboru. Každé odoslanie objednávky vymaže zo súboru predchádzajúce údaje. V tejto časti uvádzame príklad, kde sa do súboru zapisujú všetky objednávky postupne (ako pole polí). K tomu je upravený aj výpis súboru, kde sa vypisujú všetky objednávky. V kapitole 4 sme pre študentov neuvádzali túto štruktúru (pole polí), preto aj v tejto kapitole použitie tejto štruktúry štandardne neuvádzame.

Súbory `objednavka-zapis01.php` a `objednavka-zapis02.php` zapisujú do súboru údaje nie vo formáte JSON, preto nebudú fungovať s výpismi (`objednavka-vypis01---03`).

[Uloženie odoslaných údajov do súboru](#)

Funkcia `file_put_contents()` pred zapísaním údajov štandardne vymaže obsah súboru. V prípade potreby môžeme funkciu zadať ďalší parameter (konštantu `FILE_APPEND`), ktorým sa pôvodný obsah nevymaže, ale zapisované údaje sa pridajú na koniec súboru.

```
file_put_contents(sabor, udaje, FILE_APPEND)
```

[JSON](#)

Do JSON môžeme uložiť tieto typy:

- `JSONString` – textový reťazec
 - Musí byť v úvodzovkách, nie apostrofoch
- `JSONNumber` – číslo (celočíselné alebo reálne)
 - Desatinné číslo nemôže začínať bodkou, napr. `.7` nie je platný zápis, musíme napísať `0.7`

- JSONBoolean – logická hodnota
 - true, false
- JSONNull – hodnota null
- JSONArray – pole
 - Uzatvorené v hranatých zátvorkách []
 - Jednotlivé hodnoty oddeľujeme čiarkou
 - Môžeme vložiť ďalšie pole či objekt
 - Príklady:
 - [1, 2, 3]
 - [0, "PHP", false]
 - [[1, 2], null, "karta"]
- JSONObject – objekt
 - Nie je to plnohodnotný objekt, obsahuje len údaje, nie metódy.
 - Kľúče sú typu string, musia byť v úvodzovkách
 - Môžeme vkladať ďalšie objekty
 - Príklady:
 - {"x": 80, "y": 120}
 - {"osoba": {"name": "Web", "age": 25}, "login": false, "zaluby": null, "opravenia": [1,2,8]}

Užitočné zdroje a pomôcky:

- domovská stránka JSON – json.org
- JSON validátor s formátovaním kódu – jsonlint.com
- JSON Editor Online – jsoneditoronline.org
- Code Beautify (online JSON editor) – codebeautify.org/online-json-editor
- JSON editor priamo v Google Chrome – <https://chrome.google.com/webstore/detail/json-editor/lhkmoheomjkbkfloacpgllgicamhihfaj>

Zapisovanie viacerých objednávok do súboru a čítanie z tohto súboru



07/objednavka-zapis04-MP.php

PRÍKLAD

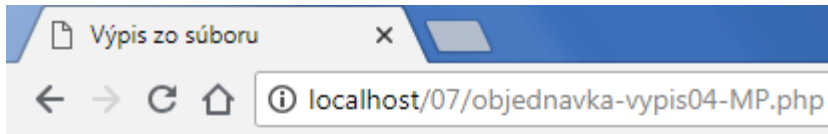
Upravme zapisovanie objednávky do súboru `objednavky.txt` tak, aby sa uchovávali všetky objednávky. Využijeme štruktúru pole polí. Tá sa nám ľahko zakóduje do JSON.

Pred zapísaním novej objednávky do súboru si najprv načítame všetky objednávky zo súboru (`$udaje_json`), dekódujeme do poľa polí (`$udaje_vsetky`), pridáme do neho pole nových údajov (`$udaje`) a až potom zapíšeme nové pole všetkých objednávok zakódované do JSON do súboru.

```
// test, či súbor existuje
if (file_exists('objednavky.txt')) {
    // načítaj pôvodné údaje zo súboru (ako JSON reťazec)
    $udaje_json = file_get_contents('objednavky.txt');
    // dekóduj JSON reťazec do poľa
```


Keďže objednávky sú číslované od 0 (počiatočný index v poli), môžeme pridať príkaz `$kluc++;`, ktorým začneme číslovať od 1.

Výpis môže vyzeráť nasledovne:



Zoznam objednávok

Objednávka č. 0

meno: Janko Hraško
adresa: Halušková 10,
Hraškovce
tovar: Made Zigy (38.9 €)

Objednávka č. 1

meno: IT Akadémia
adresa: FMFI UK,
Mlynská dolina,
Bratislava
tovar: LEGO Boost 17101 (141.9 €)

Objednávka č. 2

meno: Jožko Mrkvička

8 PHP - PAMÄTANIE ÚDAJOV

V predchádzajúcich kapitolách sme vytvorili základ internetového obchodu s rôznymi vylepšeniami. Ukázali sme si, ako ukladať údaje z objednávky, ale doteraz sme neriešili tzv. košík - teda možnosť priebežne si budovať objednávku pomocou nákupného košíka, ktorý až na záver odošleme. Budovanie a prácu s košíkom si ukážeme v tejto kapitole. A takisto, ako tento mechanizmus využiť aj iným spôsobom.

Pamätanie si údajov v košíku vlastne znamená, že údaje si musíme pamätať v rámci rôznych stránok - napr. na stránke *Ponuka*, *Košík*, prípadne ďalších. Toto môžeme dosiahnuť napr. pomocou tzv. **SESSION**.

V SESSION si môžeme ukladať rôzne informácie (udržiavať nejaký "stav"), napr. kto je prihlásený, obsah nákupného košíka, atď. Pri spustení SESSION na našej stránke vygeneruje PHP unikátne tzv. SESSION ID, ktoré je uložené na serveri, ale kópia je aj na strane prehliadača. Ak však už v danom prehliadači existuje SESSION ID pre danú adresu, nevygeneruje sa nové SESSION ID, ale PHP zistí, že už existuje, a tak len obnoví platnosť existujúceho ID. Ak by sme však rovnakú stránku otvorili v inom prehliadači na tom istom počítači, PHP vygeneruje iné nové SESSION ID, lebo z pohľadu PHP je to iný klient.

SESSION zaniknú zrušením alebo zatvorením prehliadača.

Pri práci so SESSION platí základné pravidlo: spustenie SESSION musí byť pred ľubovoľným výpisom (PHP, HTML,...). Výpisom môže byť napríklad aj prázdny riadok v zdrojovom kóde stránky (častá nepozornosť začiatočníkov).

SESSION je v skutočnosti asociatívne pole, s ktorým pracujeme ako s iným asociatívnym poľom. Pole `$_SESSION` sa vytvorí, keď spustíme príkaz `session_start()`.

Príkaz `session_start()` naštartuje (obnoví) používanie SESSION na stránke. Musí byť pred ľubovoľným výpisom (PHP alebo HTML) na stránke aj prázdny riadok v zdrojovom HTML kóde.

Po naštartovaní SESSION si môžeme do poľa `$_SESSION` ukladať údaje priradením `$_SESSION[kluc] = hodnota`.

PRÍKLAD 8.1

Uložme si do SESSION číslo pizze, ktoré bude odoslané vo formulári (`$_POST['pizza']`). Aby sme čísla pizz mali na jednom mieste, uložíme si všetky čísla do `$_SESSION['kosik']`, čo bude pole všetkých odoslaných indexov.

Pri každej pizze máme jeden formulár, v ktorom je len tlačidlo `<button>`. Všetky tlačidlá (pri všetkých pizzách) sú pomenované rovnako (`name="pizza"`), ale líšia sa hodnotou (atribút `value`). Toto riešenie nám uľahčuje kontrolu odoslania formulára v tom, že je odoslané len 1 tlačidlo (vždy s rovnakým menom) a podľa hodnoty vieme zistiť konkrétnu pizzu.



08/session1/in
dex.php


```
if (isset($_POST['pizza'])) {
    $_SESSION['kosik'][] = $_POST['pizza'];
}
```

Ak by sme si chceli dočasne vypísať, čo máme uložené v SESSION, môžeme využiť funkciu `print_r($_SESSION)`.

Samozrejme, že použitie funkcie `print_r()`, je len dočasné. Obsah košíka (SESSION) si vypíšeme v peknom tvare. Budeme ho vypisovať na samostatnej stránke *Košík* (`kosik.php`).



08/session1/
kosik.php

PRÍKLAD 8.2

Vypíšme obsah nákupného košíka uloženého v SESSION (`$_SESSION['kosik']` - stačí, keď je neprázdny) ako zoznam tovarov (názvy z poľa `$nazvy`) s ich cenami (z poľa `$ceny`) na stránke `kosik.php`. Počas vypisovania tovarov si zároveň budeme počítat celkovú cenu košíka (v premennej `$celk_cena`). V prípade, že v košíku nie je uložený žiadny tovar, vypíšeme správu.

```
if (isset($_SESSION['kosik'])) {
    $celk_cena = 0;
    ?>
    <h3>V košíku máte:</h3>
    <ol>
    <?php
        foreach($_SESSION['kosik'] as $h) {
            echo "<li>{$nazvy[$h]}: {$ceny[$h]} &euro;</li>\n";
            $celk_cena += $ceny[$h];
        }
        echo "</ol>\n";
        echo "<p>Celková cena objednávky: <strong>{$celk_cena}
&euro;</strong></p>\n";
    } else {
        echo '<p>Nákupný košík je prázdny</p>';
    }
}
```

Ak by sme však uvedený kód vyskúšali, nefungoval by. Hlavným problémom je to, že chceme využívať premennú `$_SESSION`, ale nemáme naštartované SESSION. Hneď na začiatku stránky totiž musíme mať príkaz `session_start()`. Vďaka nemu sa nám sprístupnia údaje zo SESSION, ktoré sme uložili na stránke `index.php`.

```
<?php
session_start();
$nadpis = 'Nákupný košík';
...
?>
```

Výsledná stránka by mohla vyzerať napr. ako na obrázku 8.1.

ponuka	fotogaléria	rezervá
--------	-------------	---------

Nákupný košík

V košíku máte:

1. Cardinale: 3.99 €
2. Funghi: 5.05 €
3. Hawai: 4.1 €
4. Cardinale: 3.99 €

Celková cena objednávky: **17.13 €**

Obrázok 8.3 Výsledná stránka s obsahom košíka.

Ďalším krokom v našej aplikácii by mohlo byť vymazanie košíka. Či už celého alebo vymazávanie jednotlivých tovarov. Zatiaľ vyriešime ten jednoduchší variant - kompletne vymazanie košíka.

PRÍKLAD 8.3

Pridajme na stránku nákupného košíka tlačidlo s možnosťou vymazania celého obsahu košíka uloženého v SESSION.



08/session2/ko
sik.php

Aby sa nám tlačidlo zobrazilo vedľa zoznamu tovaru, vložíme kód s formulárom medzi nadpis h3 a odrážky.

```
<h3>V košíku máte:</h3>
<form method="post" class="form_kosik">
  <input type="submit" name="zrus" value="Zruš obsah košíka"
class="odosli_velke">
</form>
<ol>
```

Potrebujeme však pridať aj kód na spracovanie odoslania tohto tlačidla. Tlačidlo spracujeme ako iné odosielacie tlačidlo - otestujeme hodnotu `$_POST['zrus']` a ak existuje, tak zrušíme obsah košíka v premennej `$_SESSION['kosik']`.

```
<?php
session_start();
$nadpis = 'Nákupný košík';

if (isset($_POST['zrus'])) {
  unset($_SESSION['kosik']);
}
include('hlavicka.php');
include('navigacia.php');
?>
...
```

Je dôležité kód spracovania tlačidla na zrušenie košíka vložiť na začiatok stránky. Po kliknutí na tlačidlo sa načíta stránka od začiatku, kde zrušíme obsah košíka a následne na stránke

zobrazíme informáciu o prázdnom košíku. Prvok poľa zrušíme pomocou funkcie `unset(čosi)`. V našom prípade `unset($_SESSION['kosik']);` zruší prvok poľa. Ale táto funkcia vymaže aj jednoduchú premennú.

V tomto momente máme hotovú základnú funkčnosť nákupného košíka. Väčšina internetových obchodov však na každej stránke priebežne zobrazuje nejakú informáciu o košíku - napr. počet tovarov alebo aktuálnu cenu objednaného tovaru. Na stránke pizzérie máme pripravenú časť na zobrazovanie stručnej informácie o košíku v navigácii.



08/session2/
navigacia.php

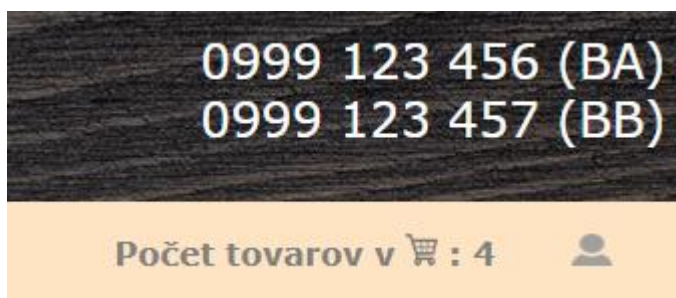
PRÍKLAD 8.4

Upravme informáciu o košíku v navigácii tak, aby sa v prípade neprázdneho košíka zobrazoval počet kusov tovaru v košíku. Využijeme funkciu `count()` pre `$_SESSION['kosik']`.

```
<div id="kosik_nav">
<?php
    if (isset($_SESSION['kosik'])) {
        $poc = count($_SESSION['kosik']);
        echo '<a href="kosik.php">Počet tovarov v  : ' . $poc . '</a>';
    } else {
        echo '<a href="kosik.php"> je
prázdny</a>';
    }
?>
<a href="">&nbsp;</a>
...
```

V kóde sme použili funkciu `count()` - vráti počet prvkov poľa.

Stručné info o košíku by mohlo vyzeráť ako na obrázku 8.2.



Obrázok 8.2 Stručná informácia o košíku v navigácii.

ÚLOHA 8.5



Doplňte príklad 8.4 tak, že na stránku košíka pridáte tlačidlo na uloženie obsahu košíka do súboru. Tlačidlo môže byť podobné tomu na vymazanie obsahu košíka. Formát údajov v súbore si zvolte podľa seba. Môžete sa inšpirovať príkladom 7.4.

8.1 Metodické pokyny pre učiteľa



CIEĽ

Cieľom tejto časti je oboznámiť študentov s možnosťou uchovávania údajov v SESSION.



VÝKLAD

Táto časť je zameraná na:

- vytváranie SESSION
- používanie a mazanie SESSION.

Výklad sa strieda s riešenými príkladmi a samostatnými úlohami pre študentov.

K session start():

Častá chyba začiatočníkov je, že keď na inej stránke pridávajú niečo do SESSION, zabudnú na začiatok súboru vložiť príkaz `session_start()`.