Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta
Katedra informatiky

WWW prezentace firmy Artex K, s. r. o.

Bakalářská práce

Vedoucí práce: PaedDr. Petr Pexa
Autor: Stanislav Kaska, DiS.

Prohlášení:

“ Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně, s použitím uvedené literatury. “

“ Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě, Pedagogickou fakultou elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách. “

V Pořežanech 17. dubna 2007 Podpis: Kaska

. . . . . .

Stanislav Kaska, DiS.

Anotace

V této bakalářské práci popisuji technologie XHTML, CSS, JavaScript, MySQL a PHP. Popisuji je, neboť jsem je využil při vytváření projektu. Projekt je zprovozněn na internetové adrese `http://www.babyobchod.cz` jako internetový obchod. Na projekt musíme nahlížet z pohledu uživatele - zákazníka a z pohledu administrátora - správce internetového obchodu. Obě tyto sekce jsou v bakalářské práci popsány.

Annotation

I describe technologies XHTML, CSS, JavaScript, MySQL and PHP in this bachelor's work. I describe these technologies because, I used them to the creation of my project. The project run on the internet address `http://www.babyobchod.cz` as internet shop. We have to regard this project as user - customer and as administrator - manager internet shop. I describe both these section in this bachelor's work.

1. Úvod

Cíl mé bakalářské práce byl zrealizovat oficiální webovou dynamickou prezentaci firmy Artex K, s. r. o. Jedná se o internetový obchod s nabídkou kompletního sortimentu pro kojence a děti zprovozněný na internetové adrese `http://www.babyobchod.cz`.

Při vytváření tohoto obchodu jsem využil technologii XHTML jako prostředek prezentace, CSS pro grafickou úpravu, JavaScript pro ošetření formulářů, MySQL databázi pro ukládání dat a PHP pro funkčnost obchodu, proto tyto technologie v této bakalářské práci popisuji.

Internetový obchod obsahuje administrátorskou sekci, kde je možnost kompletně spravovat obchod, vkládat nové produkty, vytvářet kategorie atd. Administrátorskou sekci popisuji v další části bakalářské práce. Pro uživatele jako zákazníka je možnost výběru zboží a jeho objednání, které je také popsáno. Dále uvádím část zdrojových kódu z obchodu s popisem funkčnosti.

XHTML

eXtended Hyper Text Markup Language

2.1. Co je XHTML

XHTML vychází z jazyků HTML, XML a upravených typových definic DTD. Jedná se o zpřísnění syntaktických pravidel jazyka HTML a možnost definice vlastních elementů podle pravidel jazyka XML.

U jazyka XHTML je nutné psát malými písmeny a parametry vždy do uvozovek. Nepárové elementy musí být zakončeny lomítkem / .

2.1.1. XHTML Strict

Pokud chceme využívat definici DTD Strict musíme pro vytvoření dokumentu použít jen ty nejnovější způsoby podle W3C. Definice neobsahuje žádné elementy ani atributy jazyka HTML. Grafika dokumentu je upravena pouze pomocí kaskádových stylů CSS.

```
<!DOCTYPE html
  PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

2.1.2. XHTML Transitional

V dokumentu vytvořeném pomocí DTD Transitional můžeme používat předdefinované elementy a jejich atributy, jak to umožňuje HTML, nevyužívá se však rámů.

```
<!DOCTYPE html
  PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

2.1.3. XHTML Frameset

Vytváříme-li dokument, ve kterém používáme rámy, musíme použít definici DTD Frameset. Můžeme opět používat předdefinované elementy a jejich atributu podle HTML.

```
<!DOCTYPE html
  PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

2.2. Struktura dokumentu

Takto vypadá základní kostra XHTML Transitional dokumentu. Dále si popíšeme jednotlivé části.

```
<?xml version="1.0" encoding="windows-1250"?>
<!DOCTYPE html
  PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
   <title>Název dokumentu</title>
  </head>
  <body>
   Tělo dokumentu
  </body>
</html>
```

2.3. Hlavička dokumentu <head>

Obsahuje název dokumentu a další meta elementy, které mají informativní charakter a zároveň nastavují vlastnosti dokumentu.

2.3.1. Název dokumentu

Text umístěný v elementu **<title>** se zobrazuje v záhlaví okna prohlížeče.

```
<title>Název dokumentu</title>
```

2.3.2. Meta elementy

Meta elementy využívají například vyhledávací roboti ze serveru Google nebo Morfeo pro zařazení odkazu na dokument do své databáze, díky nimž je dokument na internetu snadněji dostupnější.

```
<meta name="description" content="Popis dokumentu" />
<meta name="keywords" content="klíčová, slova, dokumentu, oddělená, čárkou" />
<meta name="author" content="Jméno autora dokumentu" />
<meta name="authormail" content="jmeno_autora@domena.cz" />
<meta name="googlebot" content="all" />
<meta name="language" content="czech" />
<meta name="country" content="cz" />
```

2.3.3. Kódování

Kódování dokumentu pro češtinu.

```
<meta http-equiv="content-type" content="text/html; charset=windows-1250" />
```

2.3.4. Ostatní

Tímto meta elementem lze zajistit přesměrování v prohlížeči na jiný dokumentu a nebo na konkrétní adresu na internetu. Přesměrování se provede po určité době (čas v sekundách).

```
<meta http-equiv="refresh" content="7; url=http://www.kam_cheme_prejit.cz" />
```

Nastavení ikonky v prohlížeči, před adresou na dokument.

```
<link href="logo.ico" rel="icon" type="image/ico" />
```

2.4. Tělo dokumentu <body>

Element `<body>` obsahuje následující atributy a pomocí jejich hodnot lze nastavit vlastnosti a vzhled dokumentu. Mezi tagy `<body>` a `</body>` vkládáme obsah dokumentu, který je popsán dále.

atribut:	hodnota:	popis:	příklad:
background	cesta k souboru	obrázek na pozadí dokumentu	<body **background**="obrazek.jpg">
bgcolor	barva	barva pozadí dokumentu	<body **bgcolor**="blue">
text	barva	barva textu v dokumentu	<body **text**="red">
link	barva	barva odkazů v dokumentu	<body **link**="black">
vlink	barva	barva navštíveného odkazu v dokumentu	<body **vlink**="green">
alink	barva	barva aktivního odkazu v dokumentu	<body **alink**="lime">
leftmargin	číslo [pixely]	velikost levého okraje dokumentu	<body **leftmargin**="100">
topmargin	číslo [pixely]	velikost horního okraje dokumentu	<body **topmargin**="100">

2.5. Formátování textu

Příslušného formátování lze dosáhnout umístěním textu do odpovídajících elementů.

funkce:	výsledek:	příklad:
tučné písmo	**tučné písmo**	<**b**>tučné písmo</**b**>
písmo kurzívou	*písmo kurzívou*	<**i**>písmo kurzívou</**i**>
podtržené písmo	podtržené písmo	<**u**>podtržené písmo</**u**>
přeškrtnuté písmo	~~přeškrtnuté písmo~~	<**s**>přeškrtnuté písmo</**s**>
pevná šířka písma	pevná šířka písma	<**tt**>pevná šířka písma</**tt**>
zvětšené písmo	zvětšené písmo	<**big**>zvětšené písmo</**big**>
zmenšené písmo	zmenšené písmo	<**small**>zmenšené písmo</**small**>
dolní index	dolní $_{\text{index}}$	dolní <**sub**>index</**sub**>
horní index	horní $^{\text{index}}$	horní <**sup**>index</**sup**>
blikající písmo	blikající písmo	<**blink**>blikající písmo</**blink**>
vystředěný text	vystředěný text	<**center**>>vystředěný text</**center**>
zvýrazněný text	**zvýrazněný text**	<**strong**>zvýrazněný text</**strong**>

Pokud chceme například tučné písmo s kurzívou, správný zápis je takto:

```
<b><i>Tučný text s kurzívou</i></b>
```

2.5.1. Odstavec <p> [paragraph]

Mezi tagy `<p>` a `</p>` napíšeme text, který se bude chovat jako odstavec a lze jej zarovnat podle následujícího atributu `align` .

atribut:	hodnota:	popis:	příklad:
align	left	zarovnání odstavce doleva	<**p align**="left">
	right	zarovnání odstavce doprava	<**p align**="right">
	center	zarovnání odstavce na střed	<**p align**="center">
	justify	zarovnání odstavce do bloku	<**p align**="jistify">

2.5.2. Nadpis <h1> [header]

Text umístěný v elementu `<h1>` se bude chovat jako nadpis. Kde číslo v elementu za `h` označuje velikost nadpisu. Rozmezí je od `1` do `6`, přičemž `1` je nadpis největší úrovně. Lze použít atribut `align`, který má stejné hodnoty jako odstavec.

2.5.3. Vodorovná čára <hr /> [horizontal rule]

Nepárový element <hr /> zobrazí v dokumentu vodorovnou čáru. Přičemž použijeme-li atribut width určíme délku vodorovné čáry. Čáru lze také zarovnat pomocí atributu align .

atribut:	hodnota:	popis:	příklad:
align	left	zarovnání čáry doleva	<hr **align**="left" />
	right	zarovnání čáry doprava	<hr **align**="right" />
	center	zarovnání čáry na střed	<hr **align**="center" />
widht	číslo [pixely] nebo procenta	délka čáry	<hr **width**="50%" />
noshade	noshade	čára bez stínování	<hr **noshade**="noshade" />
size	číslo [pixely]	tloušťka čáry	<hr **size**="7" />
color	barva	barva čáry	<hr **color**="red" />

2.5.4. Písmo <font> [font]

Při aplikaci elementu <font> na text můžeme textu měnit různé vlastnosti jako např. velikost, barvu, styl písma.

atribut:	hodnota:	popis:	příklad:
size	1 až 7, +1 až +7, -7 až -1	velikost písma	<**font size**="2">Velikost písma dvě</**font**>
color	název barvy, #rrggbb	barva písma	<**font color**="lime">Zelená barva</**font**>
face	název písma	typ písma	<**font face**="verdana">Písmo Verdana</**font**>

2.5.5. Předformátovaný text <pre>

Pokud napíšeme v textu více jak jednu mezeru, na výsledek v prohlížeči se projeví jen jedna. Pokud tedy chceme zobrazit v prohlížeči více mezer lze použít element   , který představuje mezeru, a nebo lze použít element <pre> , který přesně zobrazí v prohlížeči text, jak je napsaný ve zdrojovém kódu.

```
<pre>
    N E M Ě L I   by se používat
    v textu žádné elementy!
</pre>
```

2.5.6. Konec řádky
 [break]

Umístíme-li v textu nepárový element **
,** následující text bude na nové řádce.

2.5.7. Komentář

Text umístěný v **<!-- -->** je považován za komentář a není na něj brán zřetel.

2.6. Seznamy

V XHTML můžeme použít různé druhy seznamů, jako je například číslovaný, nečíslovaný a seznam definic. Seznam obsahuje položky, lze použít i podpoložky.

2.6.1. Číslovaný seznam <ol> [ordered list]

atribut:	hodnota:	popis:	příklad:
type	a, A, i, I, 1	druh číslování	<ol type="1"> - položka jedna - položka dvě
start	X [číslo]	počáteční hodnota	<ol type="1"start="4"> - položka jedna - položka dvě

2.6.2. Nečíslovaný seznam – odrážky <ul> [unorderd list]

atribut:	hodnota:	popis:	příklad:
type	disc ◆ circle ◇ square ■	typ odrážky	<ul type="circle"> - položka jedna - položka dvě

2.6.3. Položky seznamu <li> [list item]

Můžeme použít také atribut `type` s hodnotami stejně tak, jako je to u nečíslovaného seznamu.

atribut:	hodnota:	popis:	příklad:
value	X [číslo]	hodnota položky	<ol type="1" start="4"> - položka jedna - podpoložka jedna - podpoložka dvě <li value="7">položka dvě

2.6.4. Seznam definic <dl> [definition list]

Obsahuje dva elementy. Pojem **<dt>** [defined term] a definici pojmu **<dd>**.

```
<dl>
 <dt>První termín</dt>
 <dd>Výklad prvního termínu</dd>
 <dt>Druhý termín</dt>
 <dd>Výklad druhého termínu</dd>
</dl>
```

2.7. Odkazy <a> [anchor]

Text nebo obrázek umístěný mezi tagy `<a> </a>` bude představovat odkaz, tzn. budeme moci na něj v prohlížeči kliknout a tím se dostaneme na jinou část dokumentu popř. jiný dokument. Lze jím také spustit emailového klienta s předvyplněnými údaji o příjemci.

atribut:	hodnota:	popis:	příklad:
href	cesta a název souboru #návěští mailto:jmeno@domena.cz	cíl odkazu	Úvod Skoč na třetí odstavec Uvod - část dvě webmaster@babyobchod.cz
name	libovolné jméno	jméno návěští	
title	libovolný text	popis odkazu	Úvod
target	název cílového okna _blank [nové okno]	cílové okno	Úvod

2.8. Obrázky `<img />` [image]

Nepárový element `<img />` vloží do dokumentu obrázek pomocí atributů vysvětlených v následujíc tabulce.

Jako zdrojový obrázek můžeme vkládat soubory typu **`*.gif`** , **`*.jpg`** a v novějších prohlížečích i **`*.png`** a **`*.bmp`** .

atribut:	hodnota:	popis:	příklad:
src	cesta a jméno souboru	zdrojový obrázek	
lowsrc	cesta a jméno souboru	zdrojový obrázek pro malé displeje	
alt	libovolný text	alternativní text zobrazí se v případě chybějícího zdrojového obrázku	
title	libovolný text	popis se zobrazí vedle myši při najetí na obrázek	
width	číslo [pixely], procenta	šířka obrázku	
height	číslo [pixely], procenta	výška obrázku	
align	left right	obrázek umístěn vlevo, obtékán textem vpravo; obrázek umístěn vpravo, obtékán textem vlevo	
vspace	číslo [pixely]	vertikální okraj	
hspace	číslo [pixely]	horizontální okraj	
border	číslo [pixely]	šířka rámečku okolo obrázku	
usemap	#návěští	použití klikací mapy	

Příklad:

```
<img src="kocarek.jpg" title="Kočárek" align="right" vspace="7" hspace="7" />
```

Výsledek:

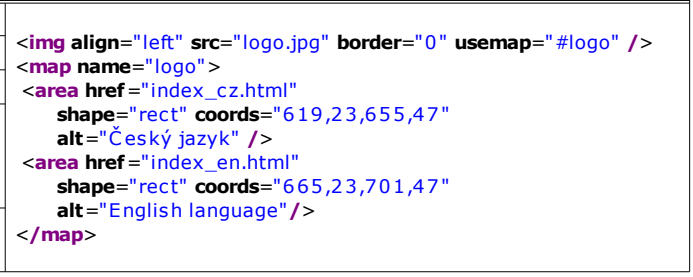
Kočár s novou lepší konstrukcí zamezující zatáčení kočáru, s regulovatelným pérováním na kolech a s přídavným bočním pérováním, přehazovací a polohovací rukojetí, vložnou taškou na miminko s připínací boudičkou, taškou na rukojeti o nosnosti 1 kg, košíkem pod kočárkem o nosnosti do 5kg, nánožníkem a s prodloužením na hluboký kočár, s odnímatelným pultíkem a pětibodovým upínáním na sportovní úpravě, s polohovací opěrkou a polohováním na nožičky. Vybaven brzdami na obou stranách. Kolečka mohou být celoplastová, plastová s nafukovacím pláštěm nebo nerezová s nafukovacím pláštěm (na obrázku plastová s nafukovacím pláštěm). Potah je z pracích atestovaných látek.

Při použití návěští `usepam` je nutno definovat oblasti obrázku, které se budou chovat jako odkazy. Vytvoření mapy `<map>` a citlivých míst `<area>` je definováno následovně:

2.8.1. Klikací mapa `<map>`

atribut:	hodnota:	popis:	příklad:
name	libovolný název	návěští uvedené v usepam	<map name="klik_mapa_01"></map>

2.8.2. Citlivá oblast `<area />`

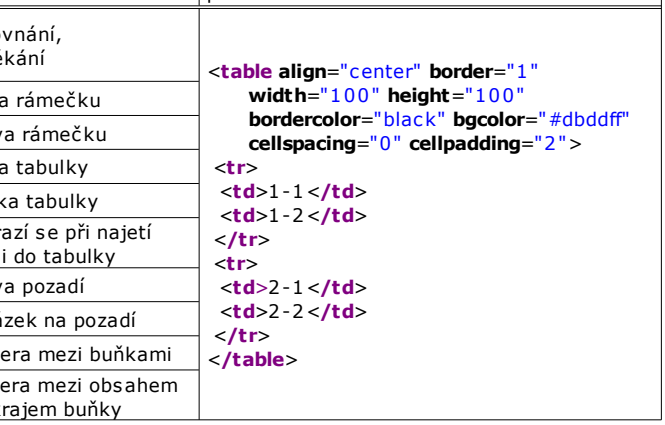
atribut:	hodnota:	popis:	příklad:
href	cesta a název souboru	cíl odkazu	
alt	libovolný text	alternativní text	
target	název cílového okna	cílové okno	
shape	defeult [celý obrázek] circle [kruh] rect [obdélník] polygon [mnohoúhelník]	vymezení citlivé oblasti	
coords	X, Y, R [kruh] X_1, Y_1, X_2, Y_2 [obdélník]	souřadnice oblasti	

```
<img align="left" src="logo.jpg" border="0" usemap="#logo" />
<map name="logo">
 <area href="index_cz.html"
     shape="rect" coords="619,23,655,47"
     alt="Český jazyk" />
 <area href="index_en.html"
     shape="rect" coords="665,23,701,47"
     alt="English language"/>
</map>
```

2.9. Tabulky

Tabulky nám napomáhají zpřehlednit obsah dokumentu. Jednotlivé buňky tabulky mohou obsahovat jakýkoliv další elementy, tzn. formátovaný text, obrázek nebo i seznam.

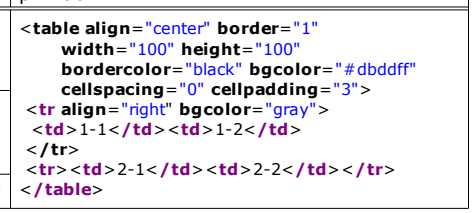
2.9.1. Tabulka `<table>`

atribut:	hodnota:	popis:	příklad:
align	left [vlevo, obtékáni vpravo] right [vpravo, obtékáni vlevo] center [na střed]	zarovnání, obtékání	
border	číslo [pixely]	šířka rámečku	
bordercolor	barva, #rrggbb	barva rámečku	
widht	číslo [pixely], procenta	šířka tabulky	
height	číslo [pixely], procenta	výška tabulky	
title	libovolný text	zobrazí se při najetí myši do tabulky	
bgcolor	barva, #rrggbb	barva pozadí	
background	cesta a jméno souboru	obrázek na pozadí	
cellspacing	číslo [pixely]	mezera mezi buňkami	
cellpadding	číslo [pixely]	mezera mezi obsahem a okrajem buňky	

```
<table align="center" border="1"
    width="100" height="100"
    bordercolor="black" bgcolor="#dbddff"
    cellspacing="0" cellpadding="2">
 <tr>
  <td>1-1</td>
  <td>1-2</td>
 </tr>
 <tr>
  <td>2-1</td>
  <td>2-2</td>
 </tr>
</table>
```

Mezi tagy `<table>` a `</table>` se musí definovat řádky `<tr>` a samotné buňky `<td>` tabulky.

2.9.2. Řádky `<tr>` [table row]

atribut:	hodnota:	popis:	příklad:
align	left [vlevo] right [vpravo] center [uprostřed]	horizontální zarovnání obsahu buněk v řádku	
valign	top [nahoru] bottom [dolu] middle [uprostřed]	vertikální zarovnání obsahu buněk v řádku	
bgcolor	barva, #rrggbb	barva pozadí řádky	

```
<table align="center" border="1"
    width="100" height="100"
    bordercolor="black" bgcolor="#dbddff"
    cellspacing="0" cellpadding="3">
 <tr align="right" bgcolor="gray">
  <td>1-1</td><td>1-2</td>
 </tr>
 <tr><td>2-1</td><td>2-2</td></tr>
</table>
```

2.9.3. Buňky <td> [table data]

atribut:	hodnota:	popis:	příklad:
align	left [vlevo] right [vpravo] center [na střed] justily [do bloku]	horizontální zarovnání obsahu v buňce	<table border="1"> <tr> <td align="center" valign="middle" width="100" height="100" colspan="2"> Nadpis</td> </tr> <tr> <td>2-1</td> <td>2-2</td> </tr> </table>
valign	top [nahoru] bottom [dolu] middle [uprostřed] baseline [k řádce]	vertikální zarovnání obsahu v buňce	
widht	číslo [pixely], procenta	šířka buňky	
height	číslo [pixely], procenta	výška buňky	
bgcolor	barva, #rrggbb	barva pozadí	
background	cesta a jméno souboru	obrázek na pozadí	
colspan	číslo	počet sloučených sloupců	
rowspan	číslo	počet sloučených řádků	
nowrap	nowrap	nezalamovat řádky	

Můžeme použít i element **<th>** [table head], který se používá namísto <tr>. Má stejné vlastnosti. Rozdíl je v tom, že text v buňce bude tučným písmem a zarovnaný na střed.

2.9.4. Nadpis tabulky <caption>

Text umístěný mezi tagy <caption> <caption> se bude zobrazovat nad tabulkou jako nadpis.

2.10. Formuláře

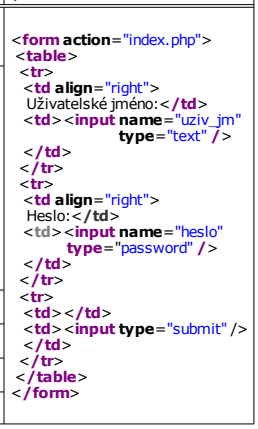
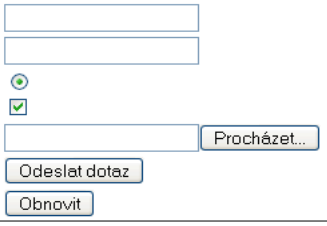
Formulář začínáme párovým elementem <form>, kde definujeme pomocí jaké metody bude formulář odesílán, jakým skriptovým souborem se vykonají vstupní data a pomocí jakého zakódování.

2.10.1. Definování formuláře <form>

atribut:	hodnota:	popis:	příklad:
action	název souboru mailto:jmeno@domena.cz	skript, který bude zpracovávat data CGI, PHP, ASP	<form action="index.php"> <form action="webmaster@babyobchod.cz">
method	post get	zabalená, odeslaná data nejsou vidět data předána jako součást adresy	v prohlížeči v řádce pro adresu uvidíme při <form action="index.php" method="post"> index.php a při <form action="index.php" method="get"> např. index.php?id=17&edit=ano
enctype	libovolná mime deklarace multipart/form-data [post] text/plain [post i get]	posílání souborů nebo použití výstupu českých znaků: multipart/form-data posílání jednoduché pošty: text/plain	*Poznámka:* V případě jednoduchého formuláře bez nahrání souboru a odeslání pošty lze atribut vynechat, jinak <form action="index.php" method="get" enctype="multipart/form-data">

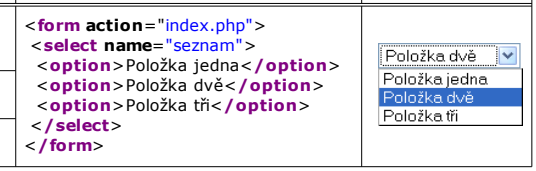
Mezi tagy <form> a <form> se musí definovat vstupní hodnoty <input />, posuvný seznam <select> nebo textové pole <textarea> .

2.10.2. Vstupní hodnoty `<input />`

atribut:	hodnota:	popis:	příklad:
name	libovolné jméno	povinné jméno prvku název proměnné ve skriptu	
type	hidden [nezobrazuje se] text [jeden řádek textu] password [heslo] radio [přepínač] checkbox [výběr] file [soubor] submit [odešle formulář] button [uživatelské tlačítko] reset [nastavení výchozí hodnoty]	
value	číslo nebo text	počáteční hodnota	
checked	checked	přednastavení přepínače a výběru	
size	číslo	šířka textového pole	
maxlenght	číslo	maximální počet znaků	

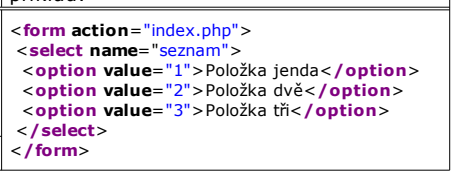
```
<form action="index.php">
 <table>
  <tr>
   <td align="right">
   Uživatelské jméno:</td>
   <td><input name="uziv_jm"
              type="text" />
   </td>
  </tr>
  <tr>
   <td align="right">
   Heslo:</td>
   <td><input name="heslo"
        type="password" />
   </td>
  </tr>
  <tr>
   <td></td>
   <td><input type="submit" />
   </td>
  </tr>
 </table>
</form>
```

2.10.3. Posuvný seznam `<select>`

atribut:	hodnota:	popis:	příklad:	ukázka:
name	libovolný text	povinné jméno prvku název proměnné ve skriptu		
size	číslo	počet zobrazených řádků		
multiple	multiple	možnost hromadného výběru		

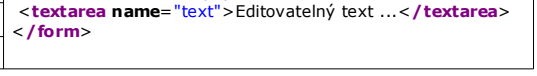
```
<form action="index.php">
 <select name="seznam">
  <option>Položka jedna</option>
  <option>Položka dvě</option>
  <option>Položka tři</option>
 </select>
</form>
```

2.10.4. Položky seznamu `<option>`

atribut:	hodnota:	popis:	příklad:
value	libovolný text nebo číslo	tato hodnota bude přiřazena proměnné ve scriptu uvedeném v atributu action v elementu <form> s názvem uvedeným v atributu name v elementu <select>	
selected	selected	označení předdefinované položky	

```
<form action="index.php">
 <select name="seznam">
  <option value="1">Položka jenda</option>
  <option value="2">Položka dvě</option>
  <option value="3">Položka tři</option>
 </select>
</form>
```

2.10.5. Textové pole `<textarea>`

atribut:	hodnota:	popis:	příklad:
name	libovolný text	povinné jméno prvku název proměnné ve skriptu	
cols	číslo	šířka pole	
rows	číslo	výška pole	

```
<form action="index.php">
 <textarea name="text">Editovatelný text ...</textarea>
</form>
```

CSS

Cascading Style Sheets

3.1. Co je CSS

Zkratka CSS znamená Cascading Style Sheets – kaskádové styly. CSS umožňuje zpracovat grafickou úpravu dokumentu prezentovaného na internetu a oddělit tak samotná data od vzhledu.

CSS nabízí širší možnosti vzhledu dokumentu než samotné XHTML. Název kaskádové styly vyplývá z možnosti vrstvit styly.

3.2. Začlenění CSS do dokumentu

3.2.1. Přímý zápis

Používá se přímo v kódu u elementu pomocí atributu `style`. Atribut `style` lze použít u každého elementu.

```
<p style="color: blue">Tento odstavec bude modrý.</p>
```

3.2.2. Globální zápis

Pomocí stylesheet - stylopisu v hlavičce dokumentu definujeme seznam stylů. Seznam obsahuje definice, jak má být element zformátován. Stylopis píšeme mezi tagy `<style>` a `</style>`.

```
<style>
 <!--
 p { color: blue; }
 -->
</style>
```

```
<p>Odstavec bude modrý.</p>
<p>Všechny odstavce budou modré.</p>
```

Nechceme-li, aby všechny odstavce byli stejného stylu, lze to zajistit pomocí **tříd** nebo **identifikátorů.**

3.2.3. Odkazem na externí soubor

Lze použití stylopis uložený v externím souboru s koncovkou `*.css`. V dokumentu se na něj odkazujeme pomocí elementu `<link>`. Výhoda je ta, že externí soubor můžeme použít pro více dokumentů. Pouze v hlavičce stránky uvedeme element `<link>` s vhodnými parametry.

Externí soubor `styly.css` bude obsahovat text:

```
p { color: blue; }
```

Do hlavičky `<head>` stránky je nutno napsat:

```
<link href="styly.css" rel="stylesheet" type="text/css" />
```

V těle `<body>` stránky budou elementy definované v externím souboru zobrazeny podle jejich definic. V našem případě bude všechen text v odstavci `<p>` modrý.

3.3. Komentáře

Pro víceřádkový komentář začínáme text znaky `/*` a končíme `*/`. Jednořádkový komentář začínáme znaky `//`.

3.4. Základní syntaxe

CSS umožňuje nastavit každému elementu XHTML dokumentu svůj specifický styl. Styly umožňují přednastavit grafický vzhled jednotlivému elementu.

Vlastnosti elementu deklarujeme podle zápisu níže, kde na velikosti písmen nezáleží, tzn. není case-senzitive. Za elementem jsou složené závorky `{}` , ve kterých deklarujeme vlastnosti. Za jménem vlastnosti je dvojtečka `:` , za kterou je hodnota vlastnosti.

```
element { vlastnost_1: hodnota_1; vlastnost_2: hodnota_2 hodnota_3; }
```

Příklad:

Do externího souboru s názvem `styly.css` napíšeme text:

```
p {   color: blue;
      background: #EBEBEB;
      font: 13px;
      border: 1px dashed;
      margin: 3px;
      padding: 3px; }
```

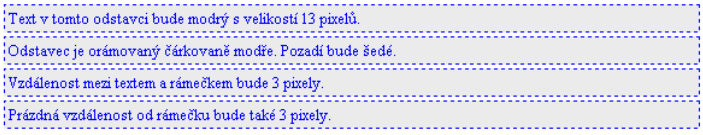
V hlavičce `<head>` stránky přilinkujeme soubor `styly.css`.

```
<link href="styly.css" rel="stylesheet" type="text/css" />
```

V těle `<body>` stránky napíšeme například následují text:

```
<p>Text v tomto odstavci bude modrý s velikostí 13 pixelů.</p>
<p>Odstavec je orámovaný čárkovaně modře. Pozadí bude šedé.</p>
<p>Vzdálenost mezi textem a rámečkem bude 3 pixely.</p>
<p>Prázdná vzdálenost od rámečku bude také 3 pixely.</p>
```

V prohlížeči bude výsledek vypadat takto:



3.4.1. Hromadná deklarace

Je možné nadefinovat vlastnosti pro více elementů najednou. Důležitá v deklaraci je čárka `,` mezi elementy. Kdyby se zde nevyskytovala, šlo by o kontextovou deklaraci.

```
element1, element2, element3 { vlastnost1: hodnota1; vlastnost2: hodnota2; }
```

```
H1, H2, H3 { color: red; } // Nadpisy 1 - 3 úrovně budou červené barvy
```

3.4.2. Kontextová deklarace

Používá se pro definování vlastností elementu vnořeného do elementu.

```
element1 element2 { vlastnost: hodnota; }
```

Příklad:

```
H4 A { font-style: italic; color: red; }
```

```
<h3>Nadpis třetí úrovně <a href="#nikam">a v něm odkaz</a></h3>
<h4>Nadpis čtvrté úrovně <a href="#nikam">a v něm odkaz</a></h4>
```

<h3>Nadpis třetí úrovně a v něm odkaz</h3>
<h4>Nadpis čtvrté úrovně a v něm odkaz</h4>

3.5. Třídy

Třídy mohou být spojeny s určitým elementem a nebo definovány samostatně. Třída se pak může použít u libovolného elementu. Třída je tedy styl podtitulu.

3.5.1. Regulární třída

Je třída, která je definovaná pouze pro určitý element. Definuje se pomocí tečky . za elementem a dále následuje název třídy, za kterým pak seznam vlastností ve složených závorkách `{}`. V dokumentu se používá atribut `class` , viz. příklad.

Definovaná třída:

```
p.ramecek {
      border: 1px dashed;
      background-color: #EEEEEE;
      margin: 3px;
      padding: 3px;
}
```

Použití v dokumentu:

```
<p>Text v tomto odstavci bude defaultní</p>
<p class="ramecek">Odstavec je orámovaný čárkovaně. Pozadí bude šedé.</p>
```

Výsledek:

Text v tomto odstavci bude defaultní.
Odstavec je orámovaný čárkovaně. Pozadí bude šedé.

3.5.2. Generická třída

Tato třída se může použít u libovolného elementu. Definuje se tečkou . a názvem třídy, dále následují uvozovky, kde je popsán seznam vlastností.

Příklad:

```
.ram {
      color: blue;
      background: #EBEBEB;
      border: 1px dashed Blue;
      margin: 3px;
      padding: 3px;
}
```

```
<p align="center">Text v tomto odstavci je defaultní, pouze zarovnaný na střed.</p>
<p class="ram">Odstavec je orámovaný čárkovaně modře. Pozadí bude šedé.</p>
<div align="center">Text v tomto elemntu je také defaultní, pouze zarovnaný na střed.</div>
<div class="ram">Odstavec v tomto elementu je orámovaný čárkovaně modře. Pozadí bude šedé.</div>
<table align="center">
 <tr><td>Buňka tabulky 1 číslo 1-1</td><td>Buňka tabulky 1 číslo 1-2</td><tr>
 <tr><td>Buňka tabulky 1 číslo 2-1</td><td>Buňka tabulky 1 číslo 2-2</td><tr>
</table>
<table align="center" class="ram">
 <tr><td>Buňka tabulky 2 číslo 1-1</td><td>Buňka tabulky 2 číslo 1-2</td><tr>
 <tr><td>Buňka tabulky 2 číslo 2-1</td><td>Buňka tabulky 2 číslo 2-2</td><tr>
</table>
```

Výsledek:

Text v tomto odstavci je defaultní, pouze zarovnaný na střed.

Odstavec je orámovaný čárkovaně modře. Pozadí bude šedé.

Text v tomto elementu je také defaultní, pouze zarovnaný na střed.

Odstavec v tomto elementu je orámovaný čárkovaně modře. Pozadí bude šedé.

Buňka tabulky 1 číslo 1-1	Buňka tabulky 1 číslo 1-2
Buňka tabulky 1 číslo 2-1	Buňka tabulky 1 číslo 2-2

Buňka tabulky 2 číslo 1-1	Buňka tabulky 2 číslo 1-2
Buňka tabulky 2 číslo 2-1	Buňka tabulky 2 číslo 2-2

3.6. Identifikátory

Používají se pro jednoznačný popis – identifikaci elementů, většinou se spojením s dynamickými skripty. Podstatou je jednoznačně identifikovat element. Identifikátory mají podobné vlastnosti jako třídy, ale obvykle se definují pro jeden element. Deklarace je tedy stejná jako u tříd, s tím že na začátku se místo tečky **.** píše mříž **#** .

```
#idnazev { vlastnost: hodnota; }
```

V dokumentu se k elementu přiřazuje atribut **id**.

Příklad:

```
#ohraniceni {
      margin: 5px;
      padding: 5px;
      border: 1px ridge;
      font: 13px Verdana;
}
```

```
<center id="ohraniceni">Text pouze v elementu <code>&lt;center&gt;</code></center>
```

Text pouze v elementu

3.7. Předdefinované prvky

V CSS se vyskytují pseudoelementy, kterými lze definovat speciální vlastnosti elementu.

3.7.1. Pseudotřídy

Defaultně se odkazy v dokumentu zobrazují podtrženě modře. Při kliknutí na některý se změní barva na červenou . Po znovu načtení dokumentu se navštívený odkaz zobrazuje purpurově. Pokud chceme, aby tomu tak nebylo, lze využít následující pseudotřídy:

a:link
a:visited - již navštívený odkaz
a:active - při kliknutí, aktivní
a:hover - při pohybu myší nad ním

Příklad:

```
a { text-decoration: none; }
a:link { color: green; }
a:hover { color: black; }
a:visited { color: silver; }
```

```
<a href="#neco">odkaz</a>
```

Zobrazený poprvé: odkaz
Po najetí myši: odkaz
Již navštívený: odkaz

3.7.2. Pseudoelementy

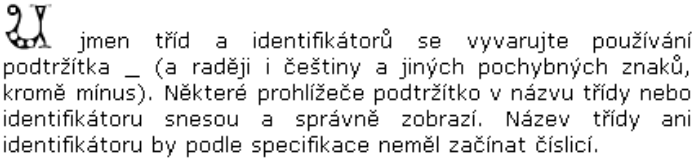
Podle typových definic se první písmeno v odstavci zvětšuje. I CSS umožňuje takovéto práce s odstavci a to následovně:

p:first-letter - velikost prvního písmene odstavce
p:first-line - vlastnosti první řádky odstavce

Příklad:

```
p { font: 13px Verdana; }
p:first-letter { font-size: 37px; font-family: fantasy; }
```

```
<p align="justify">
 U jmen tříd a identifikátorů se vyvarujte používání podtržítka <b>_</b>
 (a raději i češtiny a jiných pochybných znaků, kromě mínus).
 Některé prohlížeče podtržítko v názvu třídy nebo identifikátoru snesou
 a správně zobrazí. Název třídy ani identifikátoru by podle specifikace
 neměl začínat číslicí.
</p>
```



3.8. Jednotky délky

Hodnoty vlastností nastavovaných v CSS udávají většinou velikost. Hodnoty se zapisují v celých nebo desetinných číslech. Jako desetinný znak se používá tečka . . Hodnoty mohou být kladné i záporné. Ihned za hodnotou bez mezery následuje jednotka. Jednotky v CSS dělíme na absolutní a relativní.

3.8.1. Absolutní jednotky

Absolutní jednotky jsou přesně definovány. Jsou odvoditelné od jednotky metr [m], která je v tabulce SI. Přehled jednotek s popisem je v následují tabulce.

značka	název	převod	přepočet na pixely
cm	milimetr	1 mm = 0,1 cm	1 mm = 3,78 px
cm	centimetr	1 cm = 10 mm	1 cm = 37,8 px
in	palec	1 in = 2,54 cm = 72 pt = 6 pc	1 in = 96 px
pt	typografický bod	1 in = 1/72 in = 1/12 pc	3 pt = 4px
pc	pica nebo cicero	1 pc = 12 pt = 1/6 in	1 pc = 16 px

Jednotky `pt` a `pc` se odvozují od anglického palce. V běžném životě se s nimi lze setkat. Například `pt` je jednotka, ve které se udává velikost písmen v textových editorech.

Poznámka: Převod jednotek na pixely je závislý na nastavení systémového zobrazení. Normálně jsou monitory nastavené na 96 dpi (96 px na palec). Uživatel si může dpi (dotted per inch) monitoru změnit (např. ve Windows takto: *Ovládací panely* > *Zobrazení* > *Nastavení* > *Upřesnit* > *Obecné* > *Nastavení dpi,* - 100% pak odpovídá 96 px).

3.8.2. Relativní jednotky

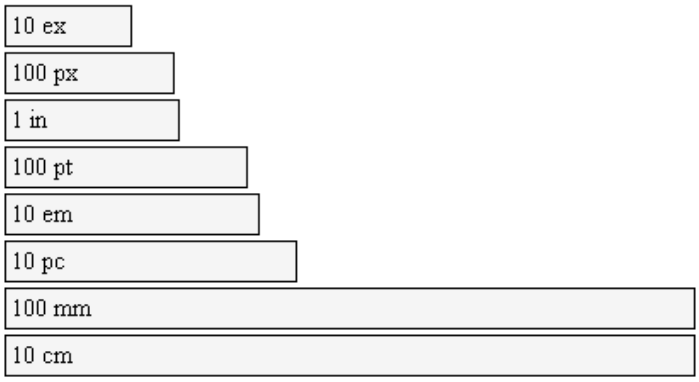
Hodnota relativních jednotek není přesně definovaná a závisí na nastavení jiných parametrů. Doporučuje se používat pouze pixel `px`. Jednotky `em`, `en` a `ex` jsou vztažné. Záleží na aktuální velikosti písma, které může být ovlivněno nadřazeným nastavením stylu, viz tabulka níže. Někdy jsou za relativní jednotky považovány i procenta `%` i se znaménky plus `+` a minus `-` . Například: `-17%` adt.

značka	název	vysvětlení
px	pixel	1 px je jeden obrazový bod na monitoru
em	em	šířka písmene m
en	en	šířka písmene n
ex	ex	výška písmene x

Příklad:

```
div { border: 1px solid Black;
      margin: 3px;
      font: Verdana;
      font-size: 15px;
      padding: 3px;
      background: #F5F5F5; }
```

```
<div style="width: 10ex"> 10 ex</div>
<div style="width: 100px"> 100 px</div>
<div style="width: 1in"> 1 in</div>
<div style="width: 100pt"> 100 pt</div>
<div style="width: 10em"> 10 em</div>
<div style="width: 10pc"> 10 pc</div>
<div style="width: 100mm"> 100 mm</div>
<div style="width: 10cm"> 10 cm</div>
```



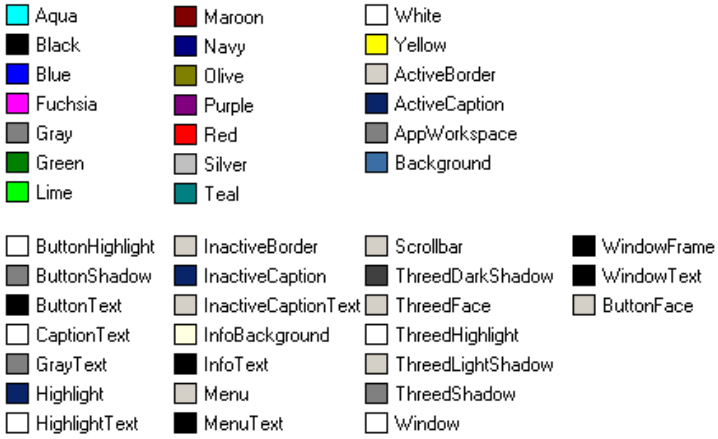
Poznámka:

Na obrázku jsou velikosti relativní a například 10 cm znázorněných na monitoru nebude 10 cm na papíře. Je zde zobrazen jen poměr jednotek.

3.9. Definice barev

Barvy lze definovat pěti způsoby:

název	definice	příklad	poznámka
anglickým jménem barvy	**color**	red, green, blue	je mnoho pojmenovaných barev
tři číslice šestnáctkové soustavy	**#rgb**	#f00, #0f0, #00f	hodnota r, g nebo b je od 0 – f
tři číslice šestnáctkové soustavy	**#rrggbb**	#ff0000, #00ff00, #0000ff	hodnota je od 00 do ff
tři čísla desítkové soustavy	**rgb(r,g,b)**	rgb(255,0,0), rgb(0,255,0), rgb(0,0,255)	hodnota čísla je od 0 do 255
procentuální vyjádření	**rgb(r%,g%,b%)**	rgb(100%,0%,0%), rgb(0%,100%,0%)	hodnota je 0 – 100 %



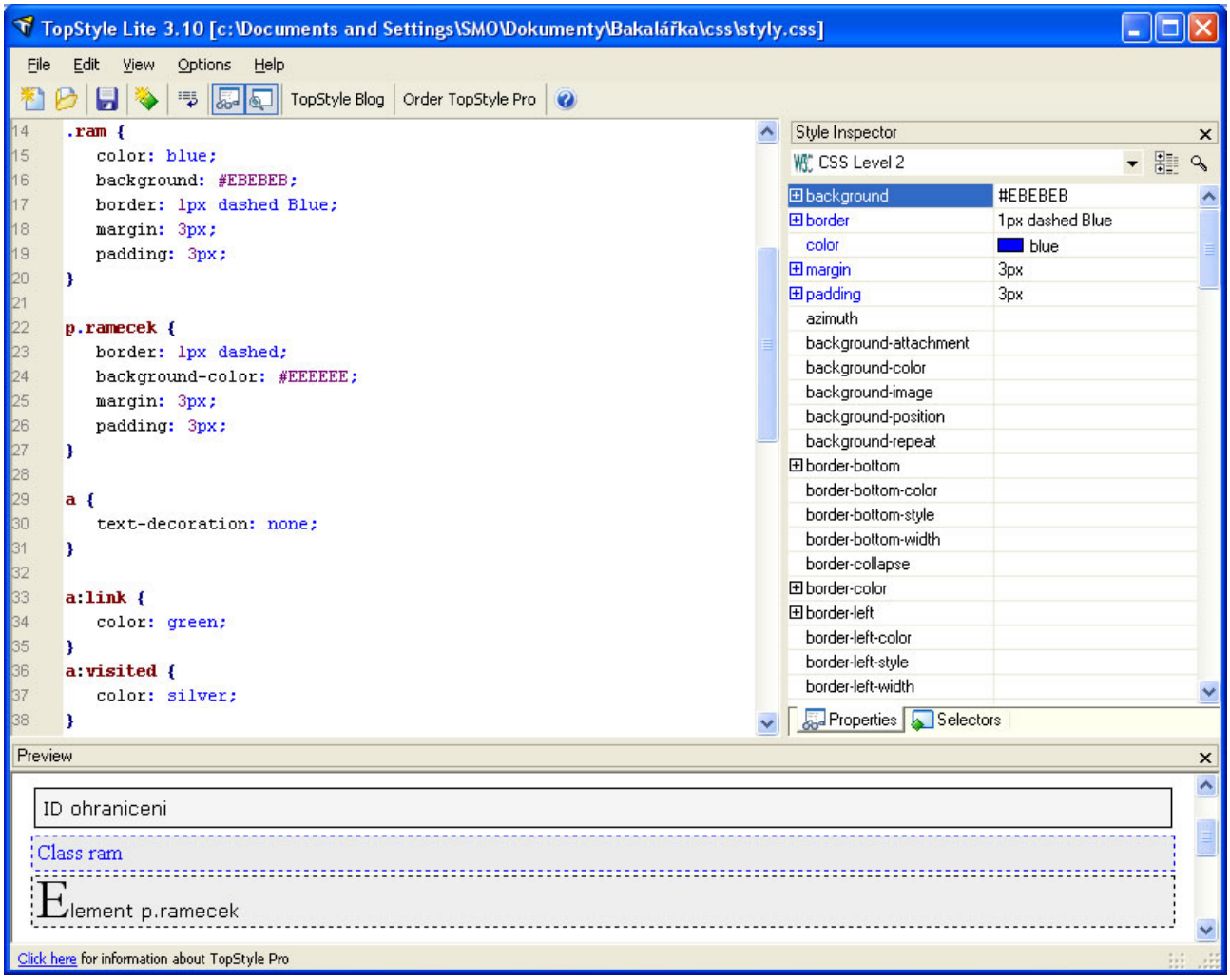
3.10. Přehled vlastností

background - vlastnosti pozadí
background-attachment - rolování pozadí
background-color - barva pozadí
background-image - obrázek na pozadí
background-position - umístění pozadí
background-repeat - opakování pozadí
border - vlastnosti rámečku
border-collapse - jednoduché rámečky v tabulkách
border-color - barva rámečku
border-style - druh rámečku
border-width - šířka rámečku
bottom - umístění od spodní části
clear - zrušení obtékání
clip - odstřihnutí objektu
color - barva objektu
cursor - kurzor myši
filter - grafické filtry
float - obtékání
font - vlastnosti písma
font-family - druh písma
font-size - velikost písma

font-style	- styl písma
font-weight	- tučnost
height	- výška prvku
left	- umístění odleva
letter-spacing	- mezery mezi znaky
line-height	- proklad řádků
list-style	- vlastnosti seznamů
list-style-image	- obrázková odrážka
list-style-position	- zastrčení odrážek do textu
list-style-type	- druh odrážek nebo číslování
margin	- okraj objektu
max-height	- maximální výška
max-width	- maximální šířka
min-height	- minimální výška
min-width	- minimální šířka
overflow	- chování obsahu, který se nevejde do rozměrů
padding	- výplň, vnitřní okraj
position	- způsob pozice prvku
right	- umístění odprava
scrollbar-x-color	- barva rolovací lišty
table-layout	- přepínání způsobu výpočtu rozměrů buněk tabulky
text-align	- zarovnání odstavce
text-decoration	- dekorace textu
text-indent	- odsazení prvního řádku
top	- umístění odshora
vertical-align	- vertikální zarovnání
visibility	- viditelnost objektu
white-space	- zacházení s řádkovými zlomy a bílými znaky
width	- šířka objektu
word-spacing	- mezery mezi slovy
writing-mode	- způsob psaní zdola nahoru
z-index	- překrývání prvků
zoom	- zvětšování

3.11. TopStyle

Pro vytváření stylů souborů `*.css` je vhodný program **TopStyle**. Skládá se ze tří částí. Vlevo můžeme psát samotný zdrojový kód CSS, který se bude přehledně barevně zobrazovat. Pokud nevíme nebo nechceme psát vlastnosti s hodnotami, můžeme si je vybrat ze seznamu vpravo. Vytvářený styl se bude zobrazovat v dolní části programu, stejně jako v prohlížeči.



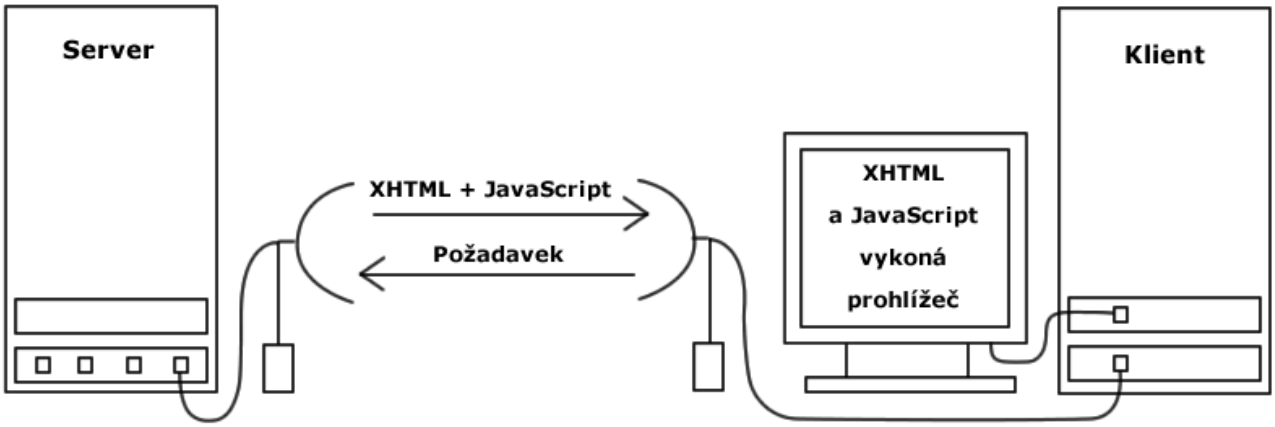
Obr. č. 3.11.1 – Program TopStyle

JavaScript

JavaScript

4.1. Co je JavaScript

JavaScript [džavaskript] je programovací jazyk, který se začleňuje přímo do XHTML kódu internetové stránky jako takzvaný klientský skript. Tento zdrojový kód ze statické internetové stránky či dynamicky vygenerované stránky, například pomocí PHP, putuje spolu s XHTML do prohlížeče klienta a zde se teprve vykonává. Opakem klientských skriptů jsou skripty serverové, které jsou vykonávány na serveru a ke klientovi putují už jen výsledky viz. PHP.



Obr. č. 4.1 - Znázornění funkce JavaScriptu

JavaScript je jazyk:
- interpretovaný, nemusí se kompilovat
- objektový, využívá objektů prohlížeče a zabudovaných objektů
- závislý na prohlížeči, funguje ve většině prohlížečů
- case senzitivní, záleží na velikosti písmen v zápisu

Omezení jazyka:
- JavaScript funguje pouze v prohlížeči
- uživatel může JavaScript zakázat
- existují různé odlišné verze jazyka i prohlížečů, což vede k častým chybám.
- neumí přistupovat k souborům (kromě cookies) ani k žádným systémovým objektům.
- neumí žádná data uložit (kromě cookies)

Cookies [kukís] jsou mechanismem na ukládání dat v prohlížeči. Slouží ke sledování nebo identifikaci vracejících se uživatelů.

4.2. Začlenění skriptu do stránky

Začlenit JavaScript do XHTML dokumentu lze třemi způsoby:

1. Začlenění přímo do dokumentu
2. Odkazem na externí soubor
3. Řádkový zápis jako atribut tagu

4.2.1. Začlenění přímo do dokumentu

```
<script language="JavaScript" type="text/javascript">
 // Začlenění JavaScriptu do HTML dokumentu
 alert("Začlenění JavaScriptu do HTML dokumentu");
</script>
```

JavaScript se může začlenit kdekoliv v XHTML kódu. Jak v hlavičce `<head>`, tak v těle `<body>` dokumentu. Prohlížeč JavaScript zpracuje okamžitě, jakmile na něj narazí.

4.2.2. Odkazem na externí soubor

```
<script language="JavaScript" type="text/javascript" src="ext_soubor.js"></script>
```

Externí soubor **ext_soubor.js**

```
// Vytiskne obsah závorky
document.write("Začlenení JavaScriptu odkazem na externí soubor");
```

JavaScript uložíme do souboru s koncovkou `*.js` . Odkaz na externí soubor vložíme na požadované místo v XHTML dokumentu, kde se vykoná. Toto se nejčastěji používá k načítání stejného JavaScriptu do více dokumentů, což je velice efektivní.

4.2.3. Řádkový zápis jako atribut elementu

Tento zápis se používá jako atribut jiného tagu, zpravidla reaguje na nějakou uživatelskou událost, např.: při přejetí objektu myší.

```
<a href="index.html" onmouseover="alert('Zpět na úvod')">Úvod</a>
```

Lze použít také řádkový zápis, který nevyužívá události a to tak, že do atributu adresy cíle napíšeme: `javascript:` a dále následuje program.

```
<a href="javascript: alert('Toto není odkaz')">Spustit script</a>
```

Po kliknutí na odkaz se spustí požadovaný JavaScript.

Nejčastější způsob použití začlenění:

- začleněním přímo do dokumentu jsou inicializovány proměnné a úvodní funkce
- odkazem na externí soubor jsou definovány funkce
- řádkovým zápisem jsou volané funkce událostmi podle rekce uživatele

4.3. Základní syntaxe

Jednotlivé příkazy se oddělují středníkem `;` nebo koncem řádky.

```
prikaz jedna; prikaz dva;
```

V názvech proměnných, v názvech objektů, všude záleží na velikosti písmen. `Prikaz` není totožné jako `prikaz`.

Řetězce se uzavírají do uvozovek `"` . Pokud se v řetězci vyskytují uvozovky můžeme použít apostrof `'` .

```
onmouseover="alert('Zpět na úvod')"
```

Použitím escape sekvence, zpětného lomítka `\` v řetězci před nějakým speciálním znakem, se tento znak vypíše do dokumentu a nebere se jako interpretovaný.

```
alet("Text s \"uvozovkami\"'");
```

Pro pravdu a nepravdu slouží speciální hodnota: pravda: **true**, nepravda: **false**

Objekty a jejich metody se oddělují tečkami **.** .

```
objekt.podobjekt.vlastnost
```

Programová sekvence se uzavírají do složených závorek **{}**.

4.4. Operátory

4.4.1. Operátory přiřazení: Slouží k nastavení hodnot proměnných.

=	přiřazení
+=	přičtení
-=	odečtení
***=**	přinásobení
/=	"přidělení"
++	zvýšení o jedničku

Totožné zápisy:

`a += 7`	`a = a + 7`
`b ++`	`b = b + 1`

4.4.2. Početní operátory: Slouží pro práci s proměnnými.

+	sčítání, spojování retězců
-	odčítáni, unární negace
*****	násobení
/	dělení

4.4.3. Logické operátory: Používají se většinou při větvení programu.

==	rovnost
!=	nerovnost
<	menší
>	větší
<=	menší a rovno
>=	větší a rovno
&&	logický AND, a zároveň
||	logický OR, nebo
? :	podmínkový výběr
,	logické spojení

4.4.4. Podmínkový výběr:

```
cislo = podmínka ? cislo_1 : cislo_2;
```

Pokud je podmínka pravdivá, `cislo` bude mít hodnotu `cislo_1` a nebude-li podmínka splněna, bude mít `cislo` hodnotu `cislo_2`.

4.5. Proměnné

Názvem proměnné může být libovolné slovo vyjma rezervovaných slov jazyka. Proměnné se deklarují klíčovým slovem `var`. Za tímto slovem následuje název proměnné. Je možno případně i přiřadit hodnotu. Není třeba deklarovat typ proměnné, JavaScript určí typ proměnné automaticky. U názvu proměnných také záleží na velikosti písmen tzv. case-sensitive.

```
var x = "Proměnná X"; // řetězec
var y = "17"; // číslo
document.write(x); // vypíše: Proměnná X
document.write("x"); // vypíše: x
```

4.5.1. Textové proměnné

U všech proměnných obsahující řetězec musí být její hodnota zadaná v uvozovkách `""` nebo v apostrofech `''` . Pokud chceme přiřadit do proměnné nějaký znak, nebo řetězec znaků, který by JavaScript chápal jako program, předřadíme před tento znak zpětné lomítko `\` .

```
odkaz = "<a href=\"index.html\">Úvod</a>";
document.write(odkaz); // Vypíše: <a href="index.html">Úvod</a>
```

nebo

```
odkaz = "<a href='index.html'>Obsah</a>";
document.write(odkaz); // Vypíše: <a href='index.html'>Úvod</a>
```

4.5.2. Logické proměnné

Pro logické hodnoty jsou rezervovaná dvě slova a to pro pravdu `true` a nepravdu `false`. Tudíž lze do proměnné uložit logickou pravdu a nebo logickou nepravdu. Hodnota se zapisuje bez uvozovek a to následovně:

```
LogPravda = true;
LogNepravda = false;
```

4.5.3. Číselné proměnné

S číselnými proměnnými se provádějí výpočty, za použití početních operátorů.

```
var y, retezec;
y = 17; y *= 3;
retezec = "Sedmnáct krát tři je: " + y;
document.write(retezec); // Vypíše: Sedmnáct krát tři je: 51
```

Script ukazuje, jak lze snadno spojit operátorem `+` textovou proměnnou s číselnou proměnnou.

4.6. Hlášky

4.6.1. Příkaz `alert()`

Dobrým efektem je příkaz `alert()`, který zobrazí upozorňovací okénko s uživatelským textem. Do závorek umístíme proměnnou nebo text v uvozovkách.

```
alert("Chceme-li v textu odřádkovat,\nslouží zpětné lomítko a písmeno: 'n'");
```

4.6.2. Příkaz `prompt()`

Příkazem `prompt()` se zobrazí dialogové okno, vyžadující vstup uživatele. Uživatel vloží hodnotu do proměnné. Hodnota může být číslo i řetězec. Zápis příkazu je následující:

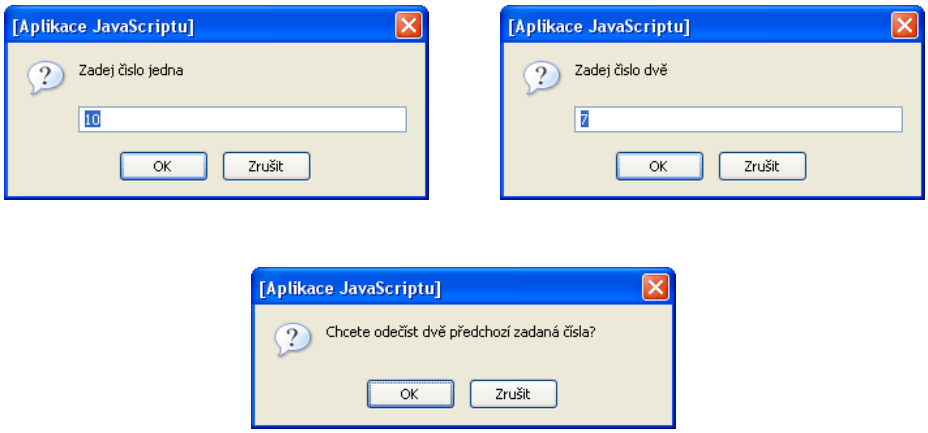
```
proměnná = prompt (hláška, implicitní hodnota);
```

```
x = prompt("Zadej číslo jedna", "10");
y = prompt("Zadej číslo dvě", "7"); z = x-y;
document.write("Rozdíl čísel " + x + " a " + y + " je: " + z);
// když x=10 a y=7 vypíše: Rozdíl čísel 10 a 5 je: 3
```

4.6.3. Příkaz `confirm()`

Příkaz `confirm()` vrací logickou hodnotu, tudíž návratová hodnota je `true` nebo `false`. Zobrazí dialogové okno s textem a výběrem `OK` nebo `Zrušit` .

```
x = prompt("Zadej číslo jedna", "10");
y = prompt("Zadej číslo dvě", "7"); z = x-y;
stav = confirm("Chcete odečíst dvě předchozí zadaná čísla?");
if(stav) document.write("Rozdíl čísel " + x + " a " + y + " je: " + z);
// Pokud potvrdíme OK vypíše: Rozdíl čísel 10 a 7 je: 3
```



Obr. č. 4.6.3.1 - Znázornění dialogových oken

4.7. Větvení a cykly

4.7.1. Příkaz `if else`

Příkaz `if else` říká, jestliže je podmínka splněna, vykonej blok příkazů, jinak vykonej jiný blok příkazů. Syntaxe příkazu `if esle` je následující:

```
if (podmínka) { příkazy, která se provedou při splnění podmínky; }
else { příkazy, které se provedou při nesplnění podmínky; }
```

Podmínka tedy může být logická proměnná. Podle hodnoty `true` nebo `false` se vykoná blok příkazu za `if` nebo za `else`. V případě, že je jen jeden příkaz za `if`, není nutno uvádět složené závorky. Příkaz `else` se nemusí uvádět.

Podmínku si můžeme vytvořit pomocí logických operátorů.

```
x = prompt("Pět plus pět je:", "55");
if (x == 10) { document.write("Správně: 5 + 5 = 10"); }
else { document.write("Špatně: 5 + 5 = 10 a ne: " + x); }
```

4.7.2. Příkaz `switch case`

Umožňuje větvení do více alternativ. Jeho syntaxe je následující:

```
switch (promenna) {
 default : príkaz_1
 case hodnota_1 : príkaz_2;
 case hodnota_2 : príkaz_3;
}
```

Podle aktuální hodnoty proměnné `promenna` se vykoná určitý příkaz. Například bude-li `promenna` mít hodnotu `hodnota_1` provede se příkaz `prikaz_2`. `Default` znamená, že se `prikaz_1` provede vždy.

4.7.3. Cyklus `while`

Blok příkazu ve složených závorkách se cyklicky vykonává, dokud platí podmínka.

```
while (podmínka) { příkazy, které se budou cyklicky opakovat; }
```

4.7.4. Cyklus `do while`

Na rozdíl od cyklu `while` se u cyklu `do while` blok příkazu vykoná minimálně jednou a poté je testována podmínka. Cyklus se provádí po dobu, je-li podmínka pravdivá.

```
do { blok příkazů; }
while (podmínka)
```

```
i = 1;
do {
 document.write(i + "<br />");
 stav = confirm("Klikni na OK a vypíše se číslo :-)"); i++;
} while (stav)
```

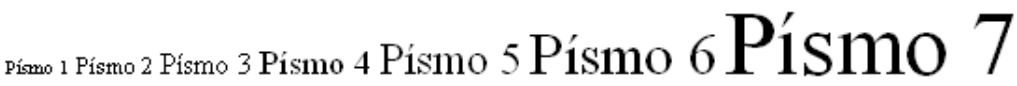
4.7.5. Cyklus `for`

V předešlých cyklech je nutno v těle cyklu zajistit změnu proměnné, která je testována v podmínce. Cyklus `for` to řeší komplexněji podle následující syntaxe:

```
for (prikaz_1; podminka; prikaz_2) { příkazy; }
```

Něž začne tělo cyklu je proveden bezpodmínečně `prikaz_1`. Dále než se provede tělo cyklu, je vyhodnocena podmínka. Pokud má hodnotu `true` , je zpracováno tělo cyklu, pokud je podmínka `false` , cyklus končí. Po provedení těla cyklu se provede `prikaz_2`. Toto je názorně zobrazeno na příkladu níže.

```
for (i=1; i <= 7; i++) {
 document.write("<font size=\""+i+"\">Písmo "+i+"</font>"); }
```



4.7.6. Cyklus `for in`

Cyklus přiřadí proměnné postupně všechny vlastnosti objektu.

```
for (podobjekt in window.screen) { document.write(podobjekt + " "); }
```

Vypíše: top left width height pixelDepth colorDepth availWidth availHeight availLeft availTop

4.7.7. Speciální příkazy pro cykly

`break` – předčasně ukončí cyklus
`continue` - skočí na začátek cyklu

4.8. Funkce

Často se ve skriptu vyskytují stejné sekvence příkazů. Funkce umožní pojmenovat takovouto sekvenci příkazů. Potom už není nutno vypisovat celý blok příkazů, stačí zavolat odpovídající funkci.

Deklarace funkce začíná v JavaScriptu vyhrazeným slovem `function` , za ním následuje jméno funkce zakončené dvojicí závorek `()`, ve kterých mohou být uvedeny parametry. Dále následuje tělo funkce, kde je napsaná určitá sekvence příkazu, která je uzavřena ve složených závorkách `{}`.

```
function jmeno_funkce(parametr_1, parametr_2) { prikaz_1; prikaz_2; prikaz_3; };
```

4.8.1. Parametry

Význam funkce je v tom, že dokáže zpracovávat parametry do funkce vstupující a reagovat na jejich hodnoty, jak ukazuje následující příklad.

```
function tisk_souctu(x,y) {                              // Deklarace funkce
 z = x + y;
 document.write(x + " + " + y + " = " + z);
};

tisk_souctu(10,15);                                      // Volání funkce
// Vytiskne: 10 + 15 = 25
```

Funkce: Jakmile prohlížeč narazí na funkci `tisk_součtu()` zjistí, jaké má parametry. V našem případě hodnoty `10` a `15` a dosadí je do proměnných, které jsou uvedeny v deklarační části v závorkách, postupně jak jdou za sebou `(x, y)`. Stejná funkce se nechá volat mnohokrát s různými parametry, vycházejí tak různé výsledky.

4.8.2. Vrácení hodnoty

Některé funkce mohou vracet hodnotu. To lze zařídit vyhrazeným slovem `return`. S takto deklarovanou funkcí musíme počítat a proto zajistit její vhodné volání, jak je uvedeno v příkladu.

Funkce se v tomto případě chová jako proměnná.

```
function mocnina(z,n) {                                    // Deklarace funkce
 var x = z;
 for (i=1; i < n; i++) { x = x * z; }
 return x;
}

x = 2; y = 4;
document.write(y + " mocnina základu " + x + " je: " + mocnina(y,x));
// Vytiskne: 4 mocnina základu 2 je: 16
```

4.8.3. Volání funkce

Jak je uvedeno v předešlých příkladech, funkci voláme pomocí námi zvoleného jména funkce a dále parametry v závorkách.

```
jmeno_funkce(hodnota_1, hodnota_2);
```

Pokud se jedná o funkci vracející hodnotu, vypadá zápis takto:

```
promenna = jmeno_funkce(hodnota_1, hodnota_2);
```

Často chceme volat funkce na základě událostí z XHTML kódu a to lze takto:

```
function upozorneni(text) { alert(text); };
```

XHTML kód:

```
<a href="index.htm"
   onclick="upozorneni('Přejděte na úvodní stránku.');">Úvodní stránka</a>
```

```
<a href="javascript: upozorneni('Toto není odkaz!')">
   Kliknutím na odkaz se vykoná pouze JavaScript</a>
```

4.8.4. Proměnné ve funkci

Proměnná deklarovaná ve funkci vyhrazeným slovem `var` je **lokální**. To znamená, že vzniká a zaniká s funkcí a není "vidět" mimo funkci. Pokud použijeme ve funkci nedeklarovanou proměnnou, pak jde o proměnou **globální**.

4.9. Atributy elementů

Pomocí takzvaných událostí můžeme měnit chování elementů. Událost může reagovat na myš, klávesnici ve formuláři, na okna, dokument prohlížeče atd. Jak je uvedeno v příkladu dále, do uvozovek za danou událost umístíme JavaScript, například změnu hodnoty nějaké vlastnosti.

4.9.1. Události myši:

onmouseout	– nachází-li se kurzor myši mimo daný element
onmouseover	– nachází-li se kurzor myši na daném elementu
onmousemove	– přejetí kurzorem myši nad elementem
onmousedown	– stisknutí tlačítka myši nad elementem
onmouseup	– uvolnění tlačítka myši
onclick	– kliknutí na element nebo při přednastavené akci
ondblclick	– při dvojitém kliknutí na element

4.9.2. Události klávesnice:

onkeydown	– stisknutí klávesy
onkeyup	– uvolnění klávesy
onkeypress	– stisknutí a uvolnění klávesy

4.9.3. Události okna a dokumentu:

onload	– při úplném načtení obsahu okna
onunload	– při odstranění obsahu dokumentu z okna
onresize	– při změně velikosti okna
onscroll	– v případě posouvání obsahu okna

4.9.4. Události formuláře:

onsubmit	– při odeslání formuláře
onreset	– v případě vymazání formuláře
onfocus	– při aktivaci políčka
onblur	– při deaktivaci políčka
onchange	– v případě změny hodnoty políčka
onselect	– v případě výběru textu v textovém poli

```
<img border="0" src="obr_1.gif" name="obr_1">

<a href="#" onmouseout="document.images['obr_1'].src='obr_1.gif'"
            onmouseover="document.images['obr_1'].src='obr_2.gif'">
 Pokud bude myš na odkazu, uvidíme obrázek dvě, jinak jedna.
</a>
```

Využitím pojmenování elementu `img` atributem `name`, tak můžeme pomocí jiného elementu `a` a použitím události `onmouseout` a `onmouseover` s vhodným spojením JavaScriptového objektu `document.images.src` , který udává hodnotu zdrojového obrázku, změnit cíl zdrojového obrázku na jiný.

4.10. Objektový pohled

JavaScript je objektově orientovaný programovací (OOP) jazyk, to znamená, že podle daného systému můžeme přistupovat ke všem vlastnostem a příkazům prohlížeče.

4.10.1. Objektový model

Je způsob, jak pojmenovat jednotlivé prvky okna prohlížeče a dokumentu tak, aby se s nimi mohlo vhodně pracovat.

4.10.1.1. Zápis

K adresování objektů se využívá tečková syntaxe. Většina objektů obsahuje podobjekty, vlastnosti nebo metody. Zápis vypadá takto:

```
objekt.podobjekt  objekt.vlastnost  objekt.metoda()
```

4.10.1.2. Terminologie

Metoda – `objekt.metoda()` – je příkaz, který něco vykonává. Je zakončena závorky `()`.

Vlastnost – `objekt.vlastnost` – nic neprovádí, obsahuje hodnotu. Podle druhu vlastnosti lze hodnotu číst nebo i měnit.

Podobjekt – `objekt.podobjekt` – může obsahovat další metody a vlastnosti.

```
<a href="javascript: window.history.back()">Zpět</a>
```

Objekt `window` obsahuje podobjekt `history`. History obsahuje metodu `back()`. Tato metoda vrací historii prohlížeče. Tento zápis funguje stejně jako tlačítko zpět v prohlížeči.

4.10.2. Základní hierarchie objektů

4.10.2.1. Objekty a podobjekty:

`window` – přístupné hlavní otevřené okno prohlížeče (pokud přistupujeme k vlastnostem nebo metodám hlavního okna, je použití `window` nepovinné)
`document` – reprezentuje obsah dokumentu v okně prohlížeče
`location` – adresa dokumentu v okně prohlížeče
`history` – obsahuje seznam navštívených adres dokumentů
`navigator` – popisuje konfiguraci prohlížeče
`event` – zpřístupnění událostí a jejich parametrů
`screen` – popisuje nastavení obrazovky
`frames` – práce s rámy
`form` – reprezentuje formuláře

4.10.2.2. Podobjekty objektu `dokument` :

`link` – odkazy na jiné dokumenty
`anchor` – obdobné jako `link`
`image` – reprezentuje obrázky umístěné na stránce
`style` – styl zobrazení elementu
`layer` – reprezentuje vytvořené vrstvy
`area` – oblasti citlivé na kliknutí či přejezd myší

4.10.2.3. Metody objektu `window` :

`open()` – otevře nové okno
`close()` – zavře okno
`moveby()` – posouvá oknem o souřadnice
`moveto()` – umístí levý horní roh okna na přesnou souřadnici obrazovky
`resizeby()` – zmenší nebo zvětší velikost okna
`resizeto()` – změní velikost okna na přesně zadané velikosti v pixelech
`scrollby()` – odroluje dokument podle zadaných souřadnic
`scrollto()` – odroluje dokument na přesnou pozici
`blur()` – přenesení okna do pozadí
`focus()` – aktivace okna, přenesení do popředí
`print()` – vytiskne aktuální okno

Příklad:

```
<a href="detail.php"
   onclick="window.open(this.href,'_blank','width=700,height=700,menubar=no,
                                            status=no,scrollbars=no');
            return false">
 Detail
</a>
```

Metoda otevře nové okno prohlížeče a v něm zadanou stránku. Okno nebude mít žádné lišty ani panely. Jeho velikost bude podle parametru `width` a `height`.

V tomto příkladu je použit objekt `this`, který ukazuje na aktuální element.

4.10.2.4. Zabudované funkce

Tyto funkce lze chápat jako metody objektu `window` .

`eval()` – vyhodnotí svoje argumenty jako program
`escape()` – zakódování řetězce
`unescape()` – rozkódování řetězce
`isfinite()` – pokud je číslo v argumentu nekonečné vrací logickou hodnotu nepravdu
`isnan()` – pokud argument není číslo vrací logickou hodnotu pravdu
`parsefloat()` – řetězec je převeden na číslo s desetinou čárkou
`parseint()` – řetězec je převeden na celé číslo

4.10.2.5. Zabudované objekty

`.length` – vrací počet znaků v řetězci
`.touppercase()` – převádí všechna písmena řetězce na velká
`.tolowercase()` – převádí všechna písmena na malá
`.toString()` – převádí čísla na řetězec
`.charat()` – podle argumentu vrátí znak z řetězce
`.charcodeat()` – vrátí unicode číslo znaku řetězce podle argumentu
`.substring()` – vrátí část řetězce od hodnoty argumentu po hodnotu druhého argumentu

MySQL

My Structured Quiry Language

5.1. Co je MySQL

MySQL je relační databázový systém šířen jako Open Source, kde každá databáze je tvořena z tabulek, které mají řádky a sloupce. Řádky obsahují jednotlivé záznamy a sloupce jména a datový typ. Práce s databázemi, tabulkami i záznamy se provádí pomocí dotazů. Dotazy vycházejí z deklarativního programovacího jazyka SQL. MySQL může být využitelné v různých programovacích jazycích například v PHP.

5.2. Databáze

Systém MySQL může obsahovat více databází. Pro práci s databázemi slouží následující dotazy.

5.2.1. Výpis databází – zobrazí jména databází

```
SHOW DATABASES;
```

5.2.2. Založení nové databáze

```
CREATE DATABASE nazev_databaze;
```

5.2.3. Nastavení aktivní databáze – nastavení jako aktivní, lze s ní pracovat

```
USE nazev_databaze;
```

5.2.4. Aktuální název databáze

```
SELECT DATABASE();
```

5.2.5. Výpis seznamu tabulek v databázi - zobrazí seznam tabulek z databáze

```
SHOW TABLES FROM nazev_databaze;
```

5.2.6. Smazání databáze – smaže celou databázi i s tabulkami

```
DROP DATABASE nazev_databaze;
```

5.3. Tabulky

Databáze může obsahovat více tabulek. Pro práci s tabulkami slouží následující dotazy.

5.3.1. Vytvoření tabulky

```
CREATE TABLE nazev_tabulky (nazev_sloupce_1 datovy_typ,
                            nazev_sloupce_2 datovy_typ);
```

Za názvem tabulky následuje závorka `( )` , kde se deklarují názvy sloupců v tabulce s příslušným datovým typem. Datové typy jsou popsány v kapitole *4. Datové typy*. Sloupců v tabulce může být i více. Pak se jednotlivé deklarace sloupců oddělují čárkou `,` .

5.3.2. Vytvoření dočasné tabulky – po uzavřeni spojení s databází tabulka zanikne

```
CREATE TEMPORARY TABLE nazev_tabulky (nazev_sloupce_1 datovy_typ,
                                      nazev_sloupce_2 datovy_typ);
```

5.3.3. Výpis popisu tabulky – zobrazí názvy sloupců, datové typy a modifikátory

```
DESCRIBE nazev_tabulky;
```

5.3.4. Změny v tabulce – provede určitou změnu s tabulkou

```
ALTER TABLE nazev_tabulky prikaz_1, prikaz_2;
```

Za názvem tabulky následuje příkaz pro změnu v tabulce. Například přidání nového sloupce, změna sloupce atd. Pokud chceme vykovat více příkazu najednou, musíme je oddělit čárkou , .

5.3.4.1. Nový sloupec

```
ADD COLUMN nazev_noveho_sloupce datovy_typ zarazeni_sloupce
```

5.3.4.1.1. Zařazení sloupce

`FIRST`	– přidá nový sloupec na začátek tabulky
`AFTER nazev_sloupce`	– přidá nový sloupec za uvedený sloupec

5.3.4.2 Smazání sloupce – odebere požadovaný sloupec

```
DROP COLUMN nazev_sloupce
```

5.3.4.3. Změna parametrů – přejmenuje sloupec a změní jeho parametry

```
CHANGE nazev_sloupce novy_nazev_sloupce novy_datovy_typ
```

5.3.4.4. Přejmenování tabulky

```
RENAME novy_nazev_tabulky
```

5.3.5. Smazání tabulky

```
DROP TABLE nazev_tabulky;
```

5.4. Datové typy

5.4.1. Celá čísla

datový typ:	rozsah hodnot:
TINYINT	-128 až 127 nebo 0 až 255
SMALLINT	-32768 až 32767 nebo 0 až 65535
MEDIUMINT	-8388608 až 8388607 nebo 0 až 16777215
INT	-2147483648 až 2147483647 nebo 0 až 4294967295
BIGINT	-9223372036854775808 až 9223372036854775807 nebo 0 až 18446744073709551615

5.4.2. Čísla s desetinnou čárkou

U reálných čísel se používají parametry **M** a **D**, kde M je omezení délky řetězce (max. 255) a D je délka čísla za desetinou čárkou (max. 30).

datový typ:	rozsah hodnot:
FLOAT(M,D)	-3.402823466E+38 až 3.402823466E+38
DOUBLE(M,D)	-1.7976931348623157E+308 až 1.7976931348623157E+308

5.4.3. Datum a čas

datový typ:	formát a rozsah:
DATE	RRRR-MM-DD, 1000-01-01 až 9999-12-31
DATETIME	RRRR-MM-DD HH:mm:SS, 1000-01-01 00:00:00 až 9999-12-31 23:59:59
TIMESTAMP(M)	RRRR-MM-DD HH:mm:SS, 1970-01-01 00:00:00 až 2036-12-31 23:59:59 M může nabývat hodnot 14, 12, 8 a 6 pak je formát RRRRMMDDHHmmSS, RRMMDDHHmmSS, RRRRMMDD, a RRMMDD v případě nevyplnění hodnoty, MySQL doplní aktuální datum a čas
TIME	HH:mm:SS, -838:59:59 až 838:59:59
YEAR(M)	RRRR, M=4 1901 až 2155, M=2 70 až 69

5.4.4. Řetězce

Délka řetězce **M** může být od 0 do 255. Vložíme-li u datového typu `CHAR` řetězec kratší než je uvedeno, chybějící znaky jsou automaticky doplněny mezerami.

datový typ:	rozsah:
CHAR(M)	0 až 255 znaků
VARCHAR(M)	0 až 255 znaků
TINYTEXT	max. 255 znaků
TEXT	max. 65535 znaků
MEDIUMTEXT	max. 16777215 znaků
LONGTEXT	max. 4294967295 znaků
ENUM(prvek1,prvek2)	max. 65535 prvků pole předem definovaných řetězců obsahuje pouze jeden prvek z definovaných
SET(prvek1, prvek2)	max. 64 prvků pole předem definovaných řetězců může obsahovat i více prvků

5.4.5. Nastavení a vlastnosti

AUTO_INCREMENT	lze použít pouze na celočíselný typ vloží unikátní číselnou hodnotu přírůstek předchozího čísla o jedničku nastavení počáteční hodnoty AUTO_INTREMENT=7
BINARY	lze použít u typ CHAR a VARCHAR datový typ bude binární a budou se rozlišovat malá a velká písmena
DEFAULT	pokud nevložíme hodnotu do buňky automaticky se doplní vychozi_hodnota DEFAULT 'vychozi_hodnota'
FULLTEXT	lze použít na typ CHAR, VARCHAR a TEXT slouží k rychlejšímu hledání řetězce v textu FULLTEXT('nazev_sloupce') při hledání použijeme příkazy MATCH a AGAINST
INDEX	umožní rychlejší přístup k datům
NOT NULL	buňka musí obsahovat nějakou hodnotu
NULL	buňka může být prázdná
PRIMARY KEY	slouží jako primární klíč identifikuje jedinečný záznam
UNIQUE	ve sloupci nesmějí být stejné hodnoty
UNSIGNED	datový typ bude bez znaménka interval datového typu se posune od 0 do X
ZEROFILL	doplní před číslo nuly do výše definované délky

5.5. Práce se záznamy

5.5.1. Vkládání záznamů – vloží řádek do tabulky

```
INSERT INTO nazev_tabulky (nazev_sloupce_1, nazev_sloupce_2)
       VALUES ('hodnota_1', 'hodnota_2');
```

Dotaz říká: Vlož nový záznam do tabulky `nazev_tabulky` do sloupce `nazev_sloupce_1` hodnotu `hodnota_1` a do sloupce `nazev_sloupce_2` hodnotu `hodnota_2`.

5.5.2. Změna záznamů – změní požadovanou hodnotu v tabulce

```
UPDATE nazev_tabulky SET jmeno_sloupce_1='nova_hodnota'
                     WHERE jmeno_sloupce_2='hodnota';
```

Dotaz říká: Změn hodnotu v tabulce `nazev_tabulky` ve sloupci `jmeno_sloupce_1` na `nova_hodnota`, kde platí podmínka že ve sloupci `jmeno_sloupce_2` je hodnota `hodnota`.

5.5.3. Výpis záznamů – zobrazí požadované záznamy z tabulky

```
SELECT pozadavky FROM nazev_tabulky WHERE nazev_sloupce='hodnota';
```

Požadavky obsahují jednotlivé názvy sloupců, které chceme vypsat. Jednotlivé sloupce oddělujeme čárkou , .Pokud chceme vypsat všechny sloupce v tabulce slouží k tomu znak hvězdička `*` .

Dotaz říká: Zobraz určité sloupce (nebo všechny) z tabulky `nazev_tabulky`, kde ve sloupci `nazev_sloupce` je hodnota `hodnota`.

5.5.3.1. Výběr z více tabulek v jeden výstup – spojí sloupce z tabulek v jeden výpis

```
SELECT * FROM tab_1 LEFT JOIN tab_2 ON
         tab_1.sloupce_z_tab_1=tab_2.sloupce_z_tab_2;
```

Je důležité, aby jeden ze sloupců z tabulky `tab_2` odpovídal hodnotám sloupce tabulky `tab_1`. Vzhledem k tomuto spojení budou přiřazeny jednotlivé záznamy z tabulky `tab_2` k záznamům tabulky `tab_1`.

Příklad:

Máme již vytvořenou tabulku `produkty` a `vyrobci`. Požadujeme výpis (imaginární tabulku), ve které budou sloupce `nazev` a `cena` z tabulky `produkty` a sloupce `jmeno` a `sidlo` z tabulky `vyrobci`. Tyto dvě tabulky budou spojeny pomocí sloupce `id_vyr` z tabulky `produkty` a sloupce `id` z tabulky `vyrobci`.

produkty

id	id_vyr	nazev	cena
1	3	stůl	2500
2	4	židle	860
3	1	postel	4500
4	2	skříň	1720

vyrobci

id	jmeno	sidlo
1	Ležíme, a. s.	Nové Město
2	KamKabaty, a. s.	Stará Lhota
3	Píšeme, s. r. o.	Česká Vesnička
4	Posaďse, s. r. o.	Stará Lhota

Použijeme na tento úkol následující dotaz:

```
SELECT nazev, cena, jmeno, sidlo FROM produkty LEFT JOIN vyrobci ON
       produkty.id_vyr=vyrobci.id;
```

Výsledek bude vypadat takto:

nazev	cena	jmeno	sidlo
stůl	2500	Píšeme, s. r. o.	Česká Vesnička
židle	860	Posaďse, s. r. o.	Stará Lhota
postel	4500	Ležíme, a. s.	Nové Město
skříň	1720	KamKabaty, a. s.	Stará Lhota

5.5.3.2. Omezení počtu řádek výpisu

```
LIMIT od,kolik;
```

Parametr `od` znamená, od kolikáté řádky se začnou výsledky vypisovat.

5.5.3.3. Seřazení sestupně, vzestupně

```
ORDER BY nazev_sloupce ASC | DESC
```

5.5.4. Mazání záznamů

```
DELETE FROM nazev_tabulky WHERE nazev_sloupce='hodnota';
```

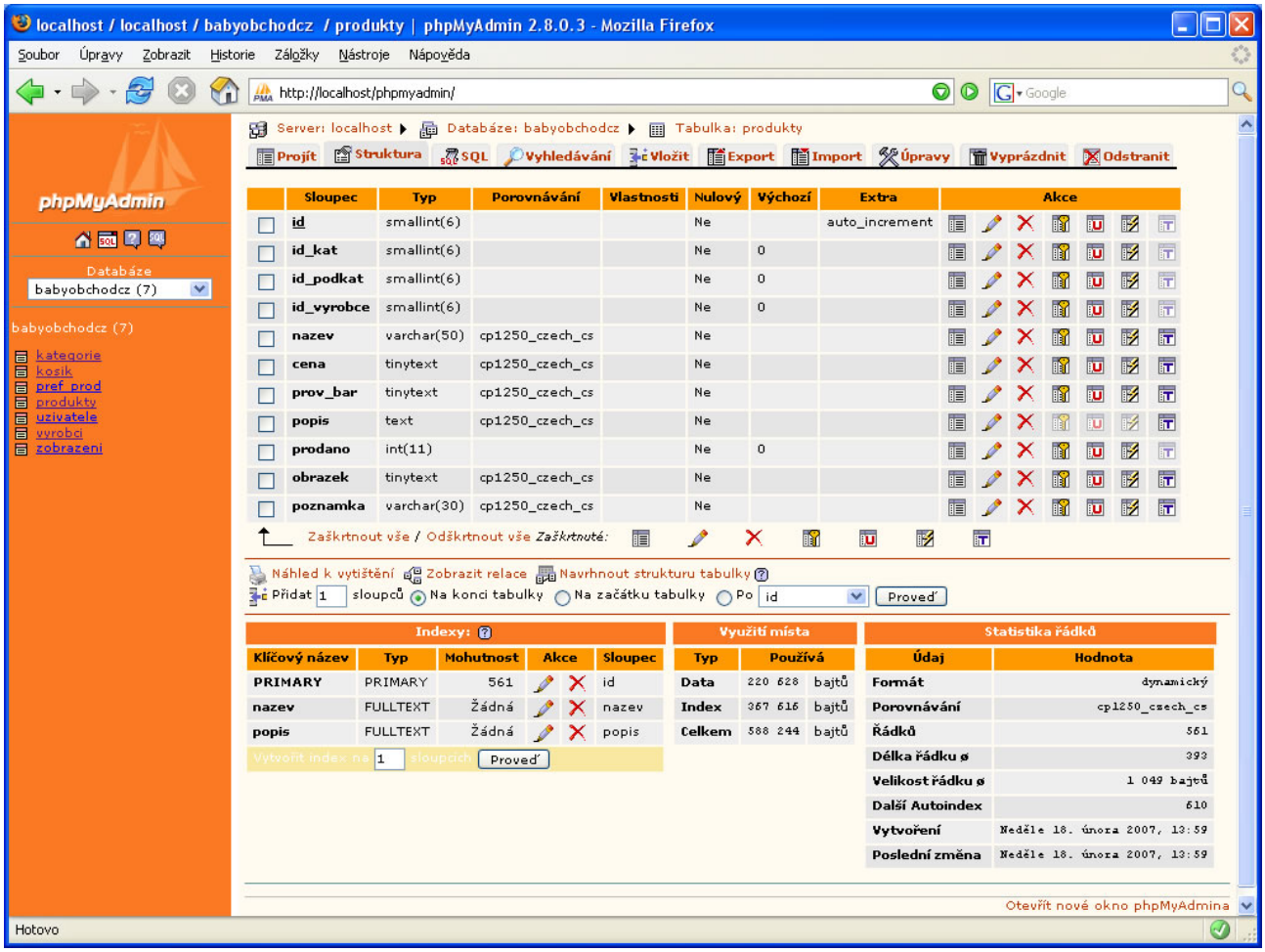
Vymaže celý řádek z tabulky.

```
TRUNCATE nazev_tabulky;
```

Smaže všechny záznamy v tabulce tím postupem, že ji smaže celou a vytvoří novou.

5.6. phpMyAdmin

Pro práci s MySQL databázemi napomáhá prostředí **phpMyAdmin**, které zpřehlední a zjednoduší práci s vytvářením tabulek, editováním záznamů, exportem a importem celých databází.



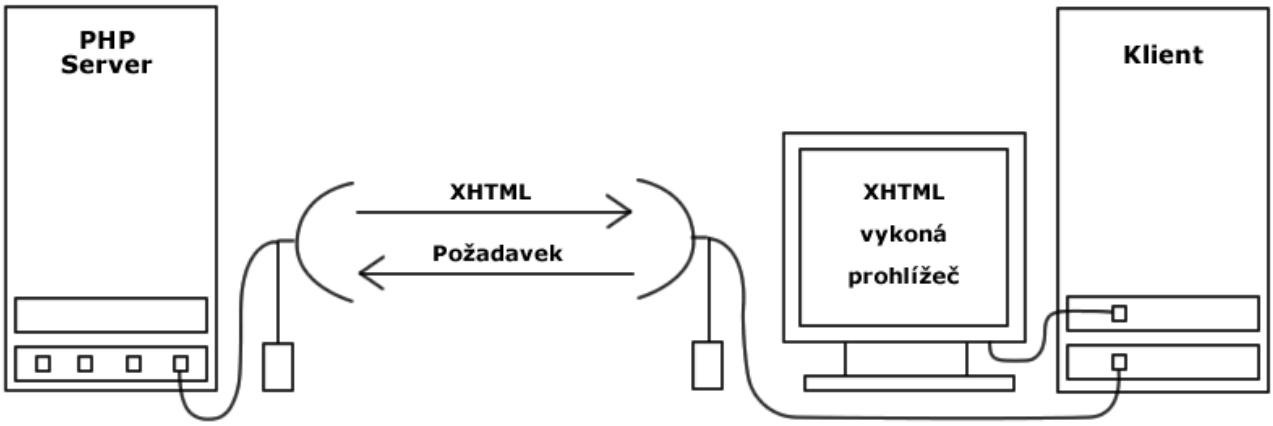
Obr. č. 5.6.1 – phpMyAdmin

PHP

Hypertext Preprocessor

6.1. Co je PHP

PHP je programovací jazyk šířen jako Open Source, který se začleňuje přímo do XHTML kódu internetové stránky jako takzvaný serverový skript. Klient na svém PC odešle požadavek na server. Po příchodu na server se požadavek zpracuje pomocí PHP skriptu začleněného do XHTML. Výsledek pouze v XHTML podobě putuje do prohlížeče klienta a zde se zobrazí.



Obr. č. 6.1 - Znázornění funkce PHP

PHP využívá podporu MySQL, tzn. mechanizmy pro práci s databázemi.

6.2. Začlenění PHP do XHTML

Pokud chceme správně začlenit PHP skript do XHTML, pak je ho nutno umístit do elementu **`<?php ?>`** . Takto napsaný dokument se zpravidla ukládá do souboru s koncovkou PHP `*.php`.

```
<?php zde_napiseme_php_skript ?>
```

6.3. Základní syntaxe

Jednotlivé instrukce se oddělují, ukončují středníkem `;` .

```
<?php instrukce_1; instrukce_2; ?>
```

6.3.1. Komentáře

Pokud chceme vložit jednořádkový komentář do PHP skriptu, musí tomuto textu předcházet dvě lomítka **`//`** .

```
<?php instrukce_1; // komentář k instrukci jedna ?>
```

Chceme-li vložit více řádkový komentář, text začínáme lomítkem a hvězdičkou **`/*`** a končíme hvězdičkou a lomítkem **`*/`** .

```
<?php
  instrukce_1;
  instrukce_2; /* je dobré dodržovat
  grafickou úpravu textu – odsazení */
?>
```

6.4. Proměnné

Názvem proměnné může být jakékoliv slovo vyjma rezervovaných slov jazyka. Proměnné nemusíme deklarovat - jsou automaticky vytvářeny při použití. I přesto, že jazyk PHP má více typů proměnných, není nutno typ deklarovat. Stačí pouze přiřadit proměnné hodnotu a to tak, že napíšeme znak dolar **$** a pak název proměnné. Dále následuje rovná se **=** a hodnota. Podle datového typu je nutno hodnotu udat do uvozovek **" "** .

```
$nazev_promene = hodnota;
```

6.4.1. Textové proměnné `string`

Hodnota textové proměnné musí být v uvozovkách **" "** a nebo v apostrofech **' '** . Pokud chceme, aby řetězec obsahoval uvozovky, vložíme řetězec do apostrofů. Pokud chceme, aby řetězec obsahoval apostrofy, vložíme řetězec do uvozovek.

Pokud chceme vložit do řetězce rezervované znaky, je nutno před tímto znakem použít znak zpětné lomítko **** .

sekvence:	význam:
\n	posun o řádek (LF, 10 v ASCII)
\r	nový řádek (CR, 13 v ASCII)
\t	tabulátor (HT, 9 v ASCII)
\\	zpětné lomítko
\$	dolar
\"	uvozovky

Řetězce lze také spojovat pomocí tečky **.** .

```
$text_promenna = "text na \"více\"\n řádcích";
$promenna = "dlouhý\n".$text_promena;
```

Proměnná `$promenna` obsahuje řetězec:

dlouhý
text na "více"
řádcích

6.4.2. Logické proměnné `boolean`

Pro logické hodnoty jsou rezervovaná dvě slova a to pro pravdu `true` a nepravdu `false`. Tudíž lze do proměnné uložit logickou pravdu a nebo logickou nepravdu. Hodnota se zapisuje bez uvozovek a to následovně:

```
$log_pravda = true;
$log_nepravda = false;
```

6.4.3. Číselná proměnná `integer`, `float`

S číselnými proměnnými lze provádět výpočty, za použití početních operátorů. Přiřazení hodnoty je bez uvozovek.

```
$a = 1234;  // kladné celé číslo
$b = -123;  // záporné celé číslo
$c = 1.234; // desetinné číslo
$d = 1.2e3; // desetinné číslo s použitím exponentu
```

6.4.4. Proměnné typu pole `array`

Proměnnou typu pole si můžeme představit jako stůl. Tento stůl má tři šuplíky. Šuplíky představují jednotlivé hodnoty proměnné pole. Určitá hodnota pole může být jakéhokoliv typu i typu pole. V šuplíku mohou být desky. Tyto desky mohou obsahovat několik listů papíru. Některý šuplík může obsahovat pouze propisky a tužky.

Pole se definuje následovně:

```
$promenna_typu_pole = array ( klic => hodnota );
```

Klíč může být pouze číslo `integer` nebo řetězec `string`. Hodnota může být jakéhokoliv typu. Pomocí klíče lze indexovat jednotlivé hodnoty v poli.

Deklarace pole pro případ výše uvedený vypadá takto:

```
$stul = array ( "suplik_01" =>
                 array ( "desky_01" =>
                          array ( "list_01" => "Seznam telefonních čísel",
                                  "list_02" => "Seznam hudby - CD",
                                  "list_03" => "Seznam filmů - DVD" ),
                         "desky_02" =>
                          array ( "list_01" => "Životopis",
                                  "list_02" => "Vysvědčení",
                                  "list_03" => "Diplom" ) ),
                "suplik_02" =>
                 array ( "desky_01" =>
                          array ( "list_01" => "PHP manuál",
                                  "list_02" => "MySQL manuál" ) ),
                "suplik_03" =>
                 array ( "propisky" => 3, "tuzky" => 2 ) );
```

Pro zkontrolování hodnot proměnné `$stul` lze použít příkaz `print_r ($stul);` , který zobrazí toto:

```
Array
(
    [suplik_01] => Array
        (
            [desky_01] => Array
                (
                    [list_01] => Seznam telefonních čísel
                    [list_02] => Seznam hudby - CD
                    [list_03] => Seznam filmů - DVD
                )
            [desky_02] => Array
                (
                    [list_01] => Životopis
                    [list_02] => Vysvědčení
                    [list_03] => Diplom
                )
        )
    [suplik_02] => Array
        (
            [desky_01] => Array
                (
                    [list_01] => PHP manuál
                    [list_02] => MySQL manuál
                )
        )
    [suplik_03] => Array
        (
            [propisky] => 3
            [tuzky] => 2
        )
)
```

Pokud chceme změnit hodnotu prvního listu v druhých deskách v prvním šuplíku stolu, zapíšeme to takto:

```
$stul["suplik_01"]["desky_02"]["list_01"] = "Curriculum Vitae";
```

Jednotlivé hodnoty pole indexujeme podle klíče. Klíč se udává v hranatých závorkách `[ ]` . Pokud se jedná o klíč typu `string` , je nutno jej uvést ještě do uvozovek. Pokud o klíč typu `integer` , uvozovky nejsou nutné.

6.5. Operátory

6.5.1. Operátory přiřazení – slouží k nastavení hodnot proměnných

operátor:	název:	příklad:
=	přiřazení	$a = 3; $b = "text";
+=	přičtení	$a += 5;
.=	spojení	$b .= "za textem";

6.5.2. Aritmetické operátory – slouží pro práci s číselnými proměnnými

operátor:	název:	příklad:
+	sčítání	$a + $b
-	odčítání	$a - $b
*	násobení	$a * $b
/	dělení	$a / $b
%	zbytek po dělení	$a % $b
++	inkrement	$a++
--	dekrement	$a--

6.5.3. Logické operátory – slouží pro práci s logickými proměnnými

operátor:	název:	příklad:	popis:
and	and, log. součin	$a and $b	pravda když $a i $b jsou true.
or	or, log. součet	$a or $b	pravda když $a nebo $b je true.
xor	xor, exkluzivní log. součet	$a xor $b	pravda když $a nebo $b je true, ale ne oba současně
!	not, negace	! $a	pravda když $a není true.
&&	and, log. součin	$a && $b	pravda když $a i $b jsou true.
||	or, log. součet	$a || $b	pravda když $a nebo $b je true.

6.5.4. Operátory porovnání – slouží k porovnání dvou hodnot

operátor:	název:	příklad:	popis:
==	rovnost	$a == $b	pravda, právě když je $a rovno $b.
====	identita	$a === $b	pravda když je $a rovno $b a navíc téhož typu
!=	nerovnost	$a != $b	pravda právě když $a není rovno $b.
<>	nerovnost	$a != $b	pravda právě když $a není rovno $b.
!==	neidentita	$a !== $b	pravda když $a není rovno $b nebo nejsou téhož typu
<	menší než	$a < $b	pravda když je $a ostře menší než $b.
>	větší než	$a != $b	pravda když je $a ostře větší než $b.
<=	menší nebo rovno	$a <= $b	pravda když je $a menší nebo rovno $b.
>=	větší nebo rovno	$a >= $b	pravda když je $a větší nebo rovno $b.

6.6. Větvení a cykly

6.5.1. Příkaz `if, elseif, else`

Příkazy říkají, jestliže je podmínka splněna, vykonej blok příkazů, jinak vykonej jiný blok příkazů. Syntaxe příkazů `if, elseif, esle` je následující:

```
if (podmínka_1) { příkazy, které se provedou při splnění podmínky jedna; }
elseif (podmínka_2) { příkazy, která se provedou při splnění podmínky jedna
                      a při splnění podmínky dvě; }
else { příkazy, které se provedou při nesplnění podmínky jedna; }
```

Podmínka může být logická proměnná, podle hodnoty `true` nebo `false` se vykoná blok příkazu za `if` nebo za `else`. Podmínku lze vytvořit i pomocí operátorů porovnání.

```
if ($a > $b) { echo "Proměnná A je větší než B"; }
else { echo "Proměnná B je větší než A"; }
// příkaz echo zobrazí v prohlížeči text umístěný v uvozovkách
```

6.6.2. Příkaz `switch`, `case`

Pokud chceme porovnávat stejnou proměnnou nebo výraz s mnoha různými hodnotami a provádět různé bloky kódu v závislosti na tom, které hodnotě se rovná, je vhodnější využít příkaz `switch` i když realizace je možná i pomocí příkazu `if`.

```
switch ($promenna) {
    case hodnota_1: prikaz_1; break;
    case hodnota_2: prikaz_2; break;
    default: prikaz;
}
```

Příkaz `break` všeobecně ukončuje provádění aktuálního příkazu `for`, `foreach`, `while`, `do while` a `switch`.

Podle aktuální hodnoty proměnné `$promenna` se vykoná určitý příkaz, např. bude-li `$promenna` mít hodnotu `hodnota_1` , provede se příkaz `prikaz_1`. Příkaz `default` znamená, že se `prikaz` provede vždy, když nevyhovuje ani jedna hodnota za `case` proměnné `$promenna`. Příkaz `default` vyhovuje všem ostatním hodnotám, které nejsou pokryty příkazem `case`.

6.6.3. Cyklus `while`

Cyklus `while` provádí vnořený blok příkazů, tak dlouho, dokud je podmínka pravdivá. Hodnota výrazu v podmínce je testována pokaždé na začátku cyklu, když se tato hodnota během provádění vnořených příkazů změní, provede se zbytek cyklu.

```
while (podmínka) { příkazy, které se budou cyklicky opakovat; }
```

Podmínka může být logická proměnná a nebo vytvořená pomocí operátorů porovnání.

6.6.4. Cyklus `do while`

Cyklus `do while` je obdobný cyklu `while` s výjimkou toho, že podmínka se testuje na konci. Cyklus se provede minimálně jednou.

```
do { blok příkazů; }
while (podmínka);
```

Použití příkazu `break` umožňuje přerušit provádění cyklu uprostřed kódu, viz. následující příklad.

```
do {
 if ($i < 7) { break; }
 echo $i;
 $i++;
} while($i < 10);
```

6.6.5. Cyklus `for`

První příkaz `prikaz_1` je proveden jednou, bezpodmínečně, na začátku cyklu. Než se provede tělo cyklu, je vyhodnocena podmínka `podminka` . Pokud je pravdivá `true` , cyklus pokračuje a zpracovává blok příkazů `blok_prikazu` , je-li nepravdivá `false` , provádění cyklu skončí. Na konci každého cyklu se provede druhý příkaz `prikaz_2` .

Každý z výrazů může být prázdný. Pokud nebude uvedena podmínka `podminka` , znamená to, že cyklus bude probíhat nekonečně dlouho. S použitím podmíněného příkazu `if` a příkazu pro ukončení cyklu `break` , lze nekonečné opakování cyklu využít.

```
for (prikaz_1; podminka; prikaz_2) { blok_prikazu; }
```

Příklad:

```
for ($i=1; $i <= 7; $i++) {
 echo "<font size=\"".$i."\">Písmo ".$i."</font>"; }
```

Výsledek:

6.6.6. Příkaz `foreach`

Proměnnou `$pole` je nadefinované pole. Při každém opakování příkazu `foreach` je hodnota pole podle aktuálního klíče přiřazena do proměnné `$hodnota`. Vnitřní ukazatel na pole – klíč je zvýšen o jedna. V příštím opakování bude proměnná `$hodnota` mít následující hodnotu pole.

Příkaz má dvě formy zápisu. Druhá forma se odlišuje tím, že je navíc aktuální hodnota klíče pole přiřazena do proměnné `$klic` .

```
foreach($pole as $hodnota) { blok_prikazu; }
foreach($pole as $klic => $hodnota) { blok_prikazu; }
```

Při prvním provedení příkazu se automaticky ukazatel – klíč nastaví na první hodnotu pole. Pokud tak chceme učinit mimo příkaz `foreach` , slouží k tomu příkaz `reset($pole);` . Tento příkaz také vrací hodnotu prvního prvku. Příkaz `foreach` pracuje s kopií pole, proto ukazatel na pole není pro jiné používání změněn.

Příklad:

```
$pole = array ("jedna" => 1, "tři" => 3, "sedm" => 7, "třináct" => 13 );
echo "Pole:<br />\n";
foreach($pole as $klic => $hodnota) {
 echo "Klíč: <b>".$klic."</b> | Hodnota: <b>".$hodnota."</b><br />\n"; }
```

Výsledek:

Pole:
Klíč: **jedna** | Hodnota: **1**
Klíč: **tři** | Hodnota: **3**
Klíč: **sedm** | Hodnota: **7**
Klíč: **třináct** | Hodnota: **13**

6.7. Funkce

Často se ve skriptu vyskytují stejné sekvence příkazů. Funkce umožní pojmenovat takovouto sekvenci příkazů. Následně není nutno příkazy vypisovat, stačí pouze zavolat odpovídající funkci.

Deklarace funkce v PHP začíná vyhrazeným slovem `function`. Za ním následuje jméno funkce, zakončené dvojicí závorek `()`, kde mohou být uvedeny argumenty, které se oddělují čárkou `,` . Dále následuje tělo funkce, kde je určitý blok příkazů, který je uzavřen ve složených závorkách `{}` .

```
function nazev_funkce ($argument_1, $argument_2) { blok_prikazu; }
```

6.7.1. Argumenty funkcí

Význam funkce je, že dokáže zpracovávat argumenty do funkce vstupující a reagovat na jejich hodnoty, viz. následující příklad.

```
function tisk_souctu($x,$y) {              // Deklarace funkce
 $z = $x + $y;
 echo $x." + ".$y." = ".$z;
};
tisk_souctu(17,13);                        // Volání funkce
// Vytiskne: 17 + 13 = 30
```

6.7.2. Návratová hodnota

Některé funkce mohou vracet hodnotu a to pomocí vyhrazeného slova **`return`**. S takto deklarovanou funkcí je nutno počítat a proto je potřeba zajistit její vhodné volání, jak se uvedeno v příkladu. Funkce se v tomto případě chová jako proměnná.

```
function mocnina($z,$n) {                                  // Deklarace funkce
 $x = $z;
 for ($i=1; $i < $n; $i++) { $x = $x * $z; }
 return $x;
}
$x = 2; $y = 4;
echo $y." mocnina základu ".$x." je: ".mocnina($y,$x); //4 mocnina základu 2 je: 16
```

6.7.3. Volání funkce

Jak je již uvedeno v předešlých příkladech, funkci lze volat pomocí námi zvoleného jména funkce a argumenty v závorkách.

```
jmeno_funkce(hodnota_1, hodnota_2);
```

Pokud se jedná o funkci vracející hodnotu, zápis vypadá následovně:

```
$promenna = jmeno_funkce(hodnota_1, hodnota_2);
```

6.7.4. Proměnné ve funkci

Jakákoli proměnná použitá uvnitř funkce je implicitně omezena jen pro vnitřní použití. Pokud požadujeme, aby takováto proměnná byla viditelná mimo funkci, je nutno před ní napsat vyhrazené slovo **`global`** – aby se chovala jako globální proměnná, například: `global $a, $b` .

6.8. Objekty a třídy

Na rozdíl od procedurálního programování, kde je uzavřena samotná část programu – funkce, je v PHP možnost programovat objektově. V objektově orientovaném programování jsou data a funkce navzájem svázány do objektů. **Třída** je pak šablona objektů. Volání třídy se jmenuje **instance**.

6.8.1. Definice třídy `class`

Objekt je definován třídou. Třída začíná klíčovým slovem `class` , dále následuje název třídy a za ním složené závorky `{}` , které ohraničují definici třídy.

```
class nazev_tridy {
 // definice třídy
}
```

Definici třídy tvoří **vlastnosti** a **metody**. Vlastnosti lze chápat jako proměnné ve třídě a metody jako funkce ve třídě.

6.8.2. Volání třídy, vlastností a metod

Proměnnou typu objekt lze vytvořit pomocí klíčového slova **`new`** , za nímž následuje název třídy i s argumenty. Definice vypadá takto:

```
$objektova_promenna = new nazev_tridy($argument);
```

Práce s objektovou proměnnou se provádí pomocí pomlčky se znakem větší `->` , za který je nutno napsat vlastnost nebo metodu přiřazené třídy a to takto:

```
$ciselna_promenna = $objektova_promenna->nazev_vlastnosti;
$objektova_promenna->nazev_metody($argument);
```

Pseudoproměnná **`$this`** odkazuje na aktuální instanci, volá se pouze uvnitř třídy.

6.8.3. Viditelnost vlastností a metod

Vlastnostem i metodám lze definovat viditelnost a to pomocí rezervovaných slov **`public`**, **`protected`** a **`private`**. Vlastnost nebo metoda definovaná slovem `public` je veřejná, přístupná odkudkoliv - ve třídě i mimo ní. Klíčovým slovem `protected` chráníme vlastnost nebo metodu. Vlastnost nebo metoda je tedy přístupná jen ve třídě samotné a nebo ve zděděné třídě. Klíčovým slovem `private` lze vytvořit soukromou vlastnost nebo metodu. Vlastnost nebo metoda je tedy přístupná pouze uvnitř třídy. V případě že není uvedeno ani jedno klíčové slovo, jedná se o veřejnou vlastnost nebo metodu.

```
class trida_viditelnosti_vlastnosti_a_metod {              // Vytvoření třídy
 public $verejna_promenna = "Veřejná proměnná";            // Vytvoření vlastností
 protected $chranena_promenna = "Chráněná proměnná";
 private $soukroma_promenna = "Soukromá proměnná";
 function volani_vlastnosti() {                            // Vytvoření metody
  echo $this->verejna_promenna;
  echo $this->chranena_promenna;
  echo $this->soukroma_promenna;
 }
}
$instance = new trida_viditelnosti_vlastnosti_a_metod();  // Vytvoření instance
echo $instance->verejna_promenna;                         // Volání vlastnosti
// echo $instance->chranena_promenna;                     // Při použití hlásí chybu
// echo $instance->soukroma_promenna;                     // Při použití hlásí chybu
$instance->volani_vlastnosti();                           // Volání metody
```

Příkaz `echo $instance->verejna_promenna;` , jímž voláme vlastnost třídy se vypíše bezproblémově. Vypíše se hodnota `Veřejná proměnná` , protože vlastnost je definovaná jako veřejná. Příkaz `echo $instance->chranena_promenna;` a `echo $instance->soukroma_promenna;` nahlásí chybu, proto jsou ve skriptu zakomentované – nelze je vlastně použít, vlastnosti nejsou viditelné. Příkaz `$instance->volani_vlastnosti();` , pomocí něhož voláme metodu třídy, vytiskne postupně hodnoty `Veřejná proměnná` , `Chráněná proměnná` a `Soukromá proměnná` , neboť metoda je veřejná a vlastnosti jsou volané uvnitř třídy. Vlastnosti jsou viditelné všechny.

6.8.4. Dědičnost

Proces vytváření nové třídy, který přebírá vlastnosti a metody jiné třídy, se nazývá dědičnost. Dědičnost třídy se provádí rezervovaným slovem `extends` . Říkáme, že původně vytvořená třída se nazývá **rodič,** nově vytvořená **potomek**. Při definování třídy jsou všechny metody rodičovské třídy přístupné pomocí zápisu **`parent::`** , za nímž následuje název rodičovské metody.

```
class nazev_tridy_potomek extends nazev_rodicovske_tridy {
 // definice třídy
 parent::nazev_rodicovske_metody();
}
```

Pokud vytváříme třídu a chceme ji zajistit tak, aby nebyla potomky přepsána, použijeme rezervované slovo **`final`** . Lze použít před rezervovaným slovem `class` nebo před názvem metody, tedy před rezervovaným slovem `function` .

```
final class nazev_tridy {
 // definice třídy;
}
```

6.8.5. Statické vlastnosti a metody

Definujeme-li vlastnost nebo metodu jako statickou, znamená to, že se váže na třídu a ne na danou instanci. Statická vlastnost nebo metoda se definuje rezervovaným slovem **`static`** , které se uvádí za definováním viditelnosti. Ke statickým vlastnostem a metodám nelze přistupovat pomocí pseudoproměnné `$this` .

Statické vlastnosti voláme zápisem `nazev_tridy::$staticka_vlastnost;` a statické metody zápisem `nazev_tridy::staticka_metoda();` . Pokud chceme přistupovat ke statickým vlastnostem a metodám uvnitř třídy, používá se místo názvu třídy rezervované slovo **`self`** . Zápis pak vypadá takto: `self::$nazev_vlastnosti;` nebo takto `self::nazev_metody();` .

6.8.6. Speciální metody

Pro třídy byly navrženy zvláštní metody, které jsou určeny pro určitou činnost. Tyto metody začínají dvěma podtržítky __ . Mezi takovéto základní metody patří konstruktor `__construct()` , destruktor `__destruct()` , `__clone` atd.

Konstruktor `__construct()` je metoda, která se vykonává při každé instanci třídy – při každém volání třídy. Metoda by měla provádět prvotní nastavení třídy.

Destruktor `__destruct()` je metoda, která se vykonává při rušení objektu. Metodu lze použít pro uvolnění paměti nebo pro ukončení spojení s databází.

Metoda `__clone` napomáhá při vytváření kopií instance. Uvnitř metody lze používat všechny vlastnosti. Kopii se vytváří pomocí zápisu `$kopie_instance = clone $instance;` .

6.9. Práce s formuláři

Vytvoříme formulář pomocí XHTML, kde v elementu `<form>` v atributu `action` bude hodnota `nazev_skriptu.php` a v atributu `method` bude hodnota `post` . Ve formuláři použijeme například element `<input>` s atributem `type` s hodnotou `text` a s atributem `name` a hodnotou `nazev_promenne` . Po odeslání formuláře, v PHP skriptu s názvem `nazev_skripu.php` v proměnné typu pole s klíčem hodnoty atributu `name` tj. `$_POST["nazev_promenne"]` , bude hodnota, kterou zadal uživatel.

Příklad: Obsah souboru: `zakodovani_hesla.php` .
Poznámka: Výstup tohoto souboru není XHTML validní, jedná se pouze o zjednodušený příklad.

```
<form action="zakodovani_hesla.php" method="post">
 Zadej heslo: <input type="password" name="heslo" />
 <input type="submit" />
</form>
<?php
 if ($_POST["heslo"] != "")
  { echo "Zakódované heslo způsobem MD5 vypadá takto:
          <b>".md5($_POST["heslo"])."</b>"; }
 else { echo "Nebylo zadáno heslo!"; }
?>
```

6.10. Práce s databází

PHP nabízí mnoho nástrojů pro práci s databázemi. Obsahuje mnoho funkcí pro práci s MySQL databází. Zde je uveden jen jednoduchý skript s MySQL databází.

Máme vytvořenou tabulku `tabulka` v MySQL databázi s názvem `jmeno_MySQL_databaze` . Tabulka obsahuje sloupce `jmeno` a `prijmeni` . Pomocí dotazu `SELECT * FROM tabulka` , který říká, vyber všechny sloupce v tabulce `tabulka` , provedeme dotaz do databáze. PHP funkcí `mysql_fetch_array()` převedeme výsledky do pole, kde klíč k hodnotě v poli má stejný název jako sloupec tabulky v databázi, jak ve uvedeno ve skriptu dále.

Příklad: Obsah souboru: `vypis_hodnoty_z_tabulky.php` .
Poznámka: Výstup tohoto souboru není XHTML validní, jedná se pouze o zjednodušený příklad.

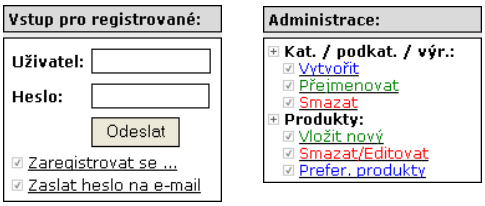
```
<?php
 $spojeni = mysql_connect("MySQL_server",                    // Vytvoří spojení
                          "MySQL_uzivatel", "MySQL_heslo");
 mysql_select_db("jmeno_MySQL_databaze");                    // Nastaví databázi
 $dotaz = "SELECT * FROM tabulka";
 $vysledek = mysql_query($dotaz);                            // Zpracuje dotaz
 $tab_vysledky = mysql_fetch_array($vysledek)
 echo "Jméno: ".$tab_vysledky["jmeno"]."<br />\n";           // Vytiskne výsledky
 echo "Příjemni: ".$tab_vysledky["jmeno"]."<br />\n";
 mysql_free_result($spojeni);                                // Uvolní paměť
 mysql_close($spojeni);                                      // Uzavře spojení
?>
```

http://www.babyobchod.cz

WWW prezentace firmy Artex K, s. r. o.

7. Administrátorská sekce

V pravé části obchodu se nachází formulář, kde se po vyplnění uživatelského jména a hesla administrátora, zobrazí skryté menu, v němž je na výběr z několika možností pro správu kategorií, výrobců a produktů.



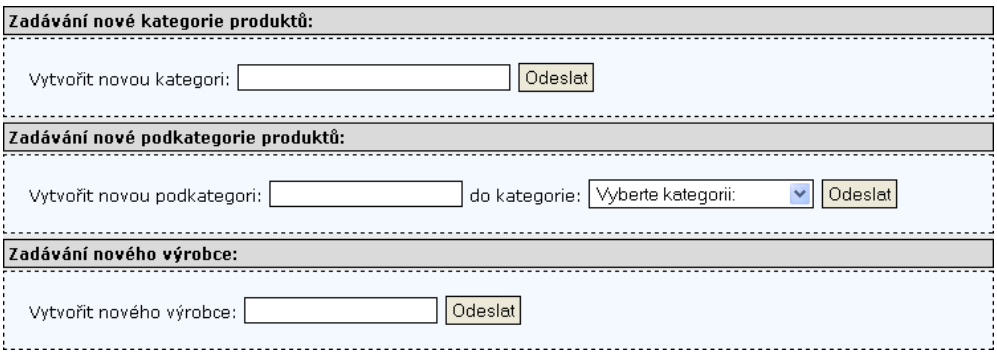
Obr. č. 7.1. – Přihlašovací formulář a administrační menu

7.1. Administrace kategorie, podkategorie a výrobce

Před vložením produktu do obchodu, je nutné vědět, do jaké kategorie a podkategorie patří a od jakého je výrobce. V případě, že požadovaná kategorie, podkategorie nebo výrobce není vytvořen, přejde se do sekce Vytvořit, kde se vytvoří. Kategorie, podkategorie nebo výrobce je také možno editovat a mazat v sekci Přejmenovat a Smazat .

7.1.1. Vytvoření

Před vytvářením podkategorie musí být vytvořena kategorie, aby ji bylo možné vybrat ze seznamu, jak je znázorněno na obrázku.



Obr. č. 7.1.1.1. – Formulář pro vytvoření kategorie, podkategorie a výrobce

7.1.2. Editace

Obr. č. 7.1.2.1. – Formulář pro editaci kategorie, podkategorie a výrobce

7.1.3. Smazat

Mazáni je vhodné provést jen v případě, že kategorie, podkategorie nebo výrobce neobsahuje žádné produkty. Pokud by tomu tak nebylo, všechny indexy v produktech odkazující se na tento název kategorie, podkategorie nebo výrobce, by byly neplatné.

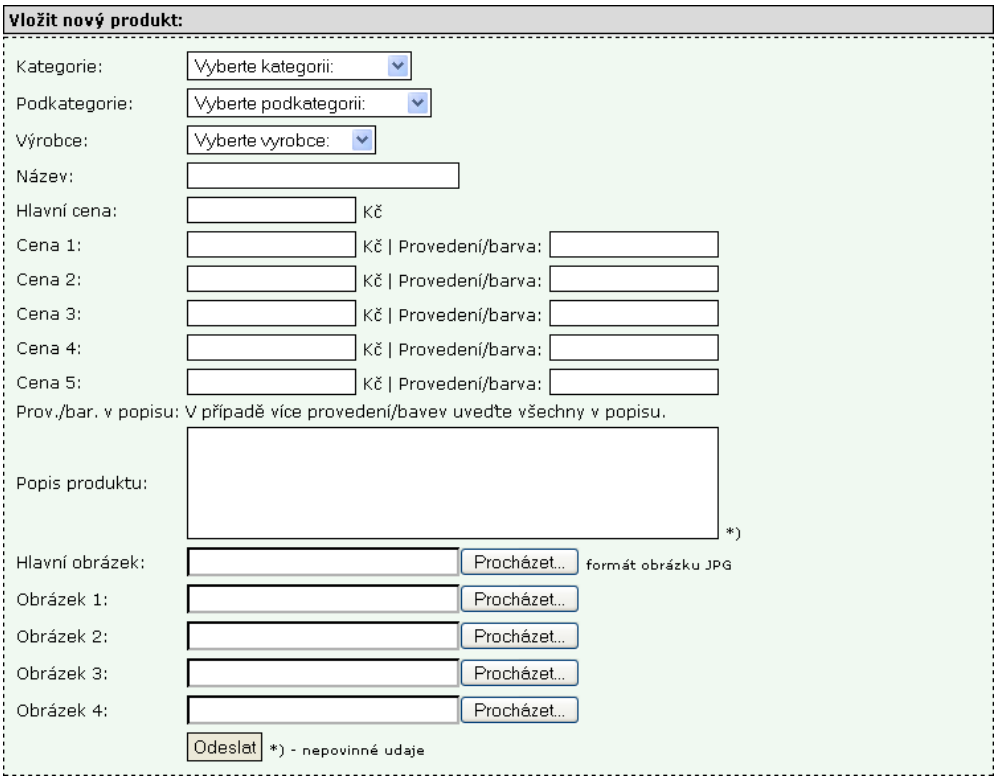
Obr. č. 7.1.3.1. – Formulář pro smazání kategorie, podkategorie a výrobce

7.2. Administrace produktů

Do databáze lze vložit nový produkt, editovat a smazat jej.

7.2.1. Vložit nový produkt

Po vytvoření kategorii, podkategorii a výrobce lze přejít k samotnému zadávání nového produktu. Ve formuláři pro zadávání produktu je nutno vybrat kategorii, podkategorie a výrobce, vyplnit název a cenu produktu. Také je povinné připojit hlavní obrázek ve formátu JPG. Popis produktu není povinný, z důvodu prodejnosti produktu, se ho doporučuje vyplnit.

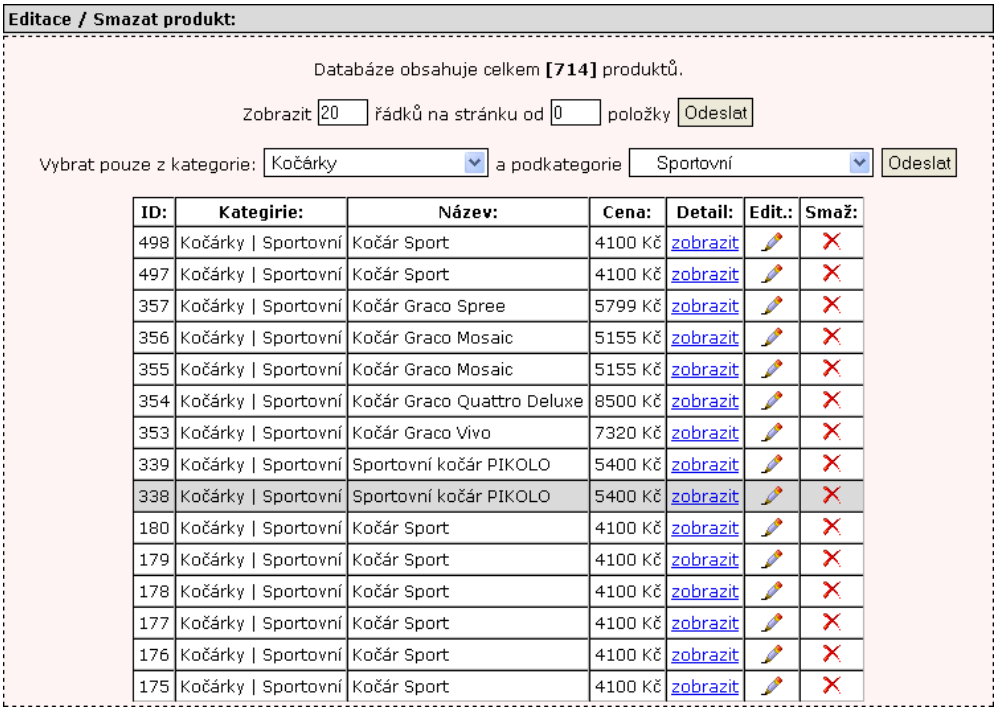


Obr. č. 7.2.1.1. – Formulář pro vytvoření nového produktu

Pokud je produkt v různém provedení nebo barvách, je na to formulář připraven. Vyplní se zde odpovídající počet provedení, kde se uvede cena a název provedení nebo barvy. Nestačí-li počet pěti provedení či barvy, uvede se v popisu. Při výběru produktu si zákazník místo zaškrtnutí odpovídajícího provedení vyplní název provedení sám, které najde v popisu. Lze také připojit více obrázků. V tomto formuláři nejvýše čtyři. Je možno i více, což se provede v sekci editace, kde se může v rámci účelnosti připojit potřebný počet obrázků. Obrázek musí být ve formátu JPG, protože pouze pro tento formát je napsán skript, který obrázek přerozměrovává. Minimální rozměry obrázku by měli být 550 x 550 pixelů.

7.2.2. Editace a mazání produktu

Aby mohl být konkrétní produkt editován nebo smazán, musí se definovat nějaký užší výběr produktů, ze kterých se vybere ten požadovaný. K tomuto slouží následující formulář, v němž se zadá počet řádků, které chceme zobrazit, od jaké položky se má výběr zobrazit a z jaké kategorie a podkategorie.



Editace / Smazat produkt:

Databáze obsahuje celkem **[714]** produktů.

Zobrazit 20 řádků na stránku od 0 položky Odeslat

Vybrat pouze z kategorie: Kočárky a podkategorie Sportovní Odeslat

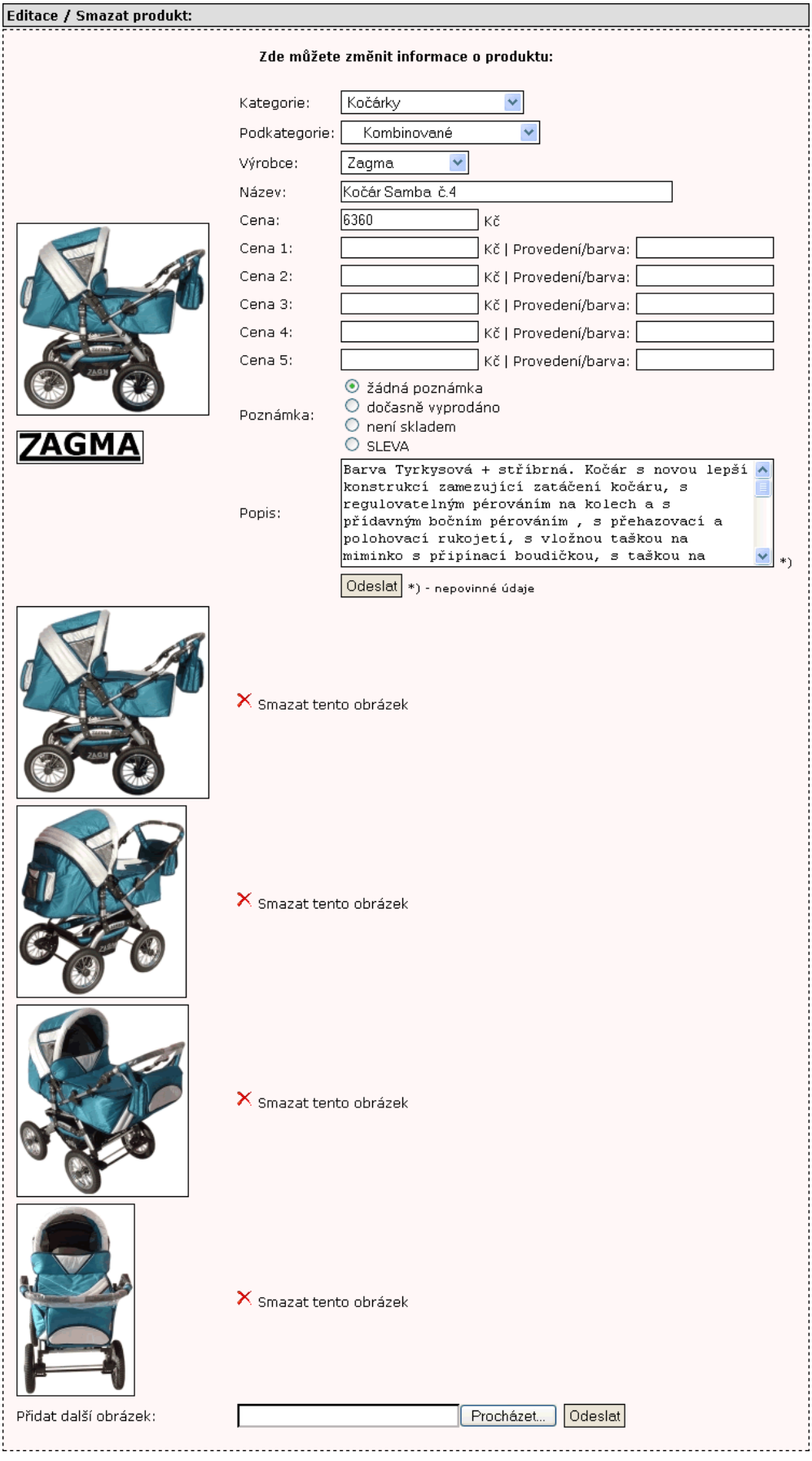
ID:	Kategirie:	Název:	Cena:	Detail:	Edit.:	Smaž:
498	Kočárky | Sportovní	Kočár Sport	4100 Kč	zobrazit		
497	Kočárky | Sportovní	Kočár Sport	4100 Kč	zobrazit		
357	Kočárky | Sportovní	Kočár Graco Spree	5799 Kč	zobrazit		
356	Kočárky | Sportovní	Kočár Graco Mosaic	5155 Kč	zobrazit		
355	Kočárky | Sportovní	Kočár Graco Mosaic	5155 Kč	zobrazit		
354	Kočárky | Sportovní	Kočár Graco Quattro Deluxe	8500 Kč	zobrazit		
353	Kočárky | Sportovní	Kočár Graco Vivo	7320 Kč	zobrazit		
339	Kočárky | Sportovní	Sportovní kočár PIKOLO	5400 Kč	zobrazit		
338	Kočárky | Sportovní	Sportovní kočár PIKOLO	5400 Kč	zobrazit		
180	Kočárky | Sportovní	Kočár Sport	4100 Kč	zobrazit		
179	Kočárky | Sportovní	Kočár Sport	4100 Kč	zobrazit		
178	Kočárky | Sportovní	Kočár Sport	4100 Kč	zobrazit		
177	Kočárky | Sportovní	Kočár Sport	4100 Kč	zobrazit		
176	Kočárky | Sportovní	Kočár Sport	4100 Kč	zobrazit		
175	Kočárky | Sportovní	Kočár Sport	4100 Kč	zobrazit		

Obr. č. 7.2.2.1. – Formulář pro výběr produktu a pro editaci nebo smazání

Pokud jsou informace o názvu produktu a zařazení v kategorii nedostačující, slouží k tomu odkaz zobrazit detail. Zobrazí se okno, kde budou celkové informace o produktu i s obrázky. Tyto informace by měli být dostačující k tomu, aby bylo jasné, co přesně editovat nebo jaký produkt smazat.

7.2.2.1. Editace produktů

Po kliknutí na ikonu tužky pro editaci, se zobrazí formulář s předdefinovanými hodnotami, kde se může změnit požadující položka, jak je znázorněno na obrázku dále. Možnost změnit cenu, popis, zadat poznámku o stavu produktu atd. Je-li skladem, je-li vyprodáno nebo je-li výrobek ve slevě. Možnost smazat již vložený obrázek a nebo naopak přidat další.



Obr. č. 7.2.2.1.1. – Formulář pro editaci produktu

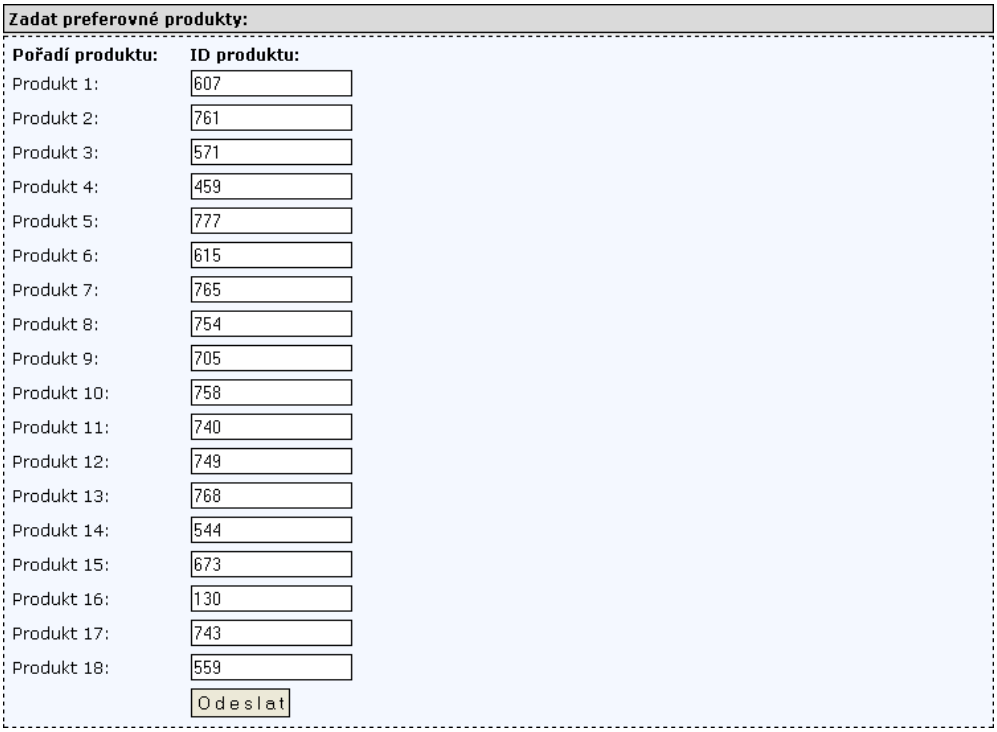
7.2.2.2. Smazání produktu

Po kliknutí na ikonu červeného přeškrtnutí pro smazání se zobrazí výzva se základními informacemi o produktu. Po odeslání bude produkt smazán.

Obr. č. 7.2.2.2.1. – Výzva k smazání produktu

7.3. Preferované produkty

Jednoduchým formulářem, kde je pouze číslo produktu a ID (identifikační číslo produktu), se zajistí, jaké produkty a v jakém pořadí se budou zobrazovat při spuštění úvodní stránky nebo pouze při prvním otevření obchodu `http://www.babyobchod.cz` .



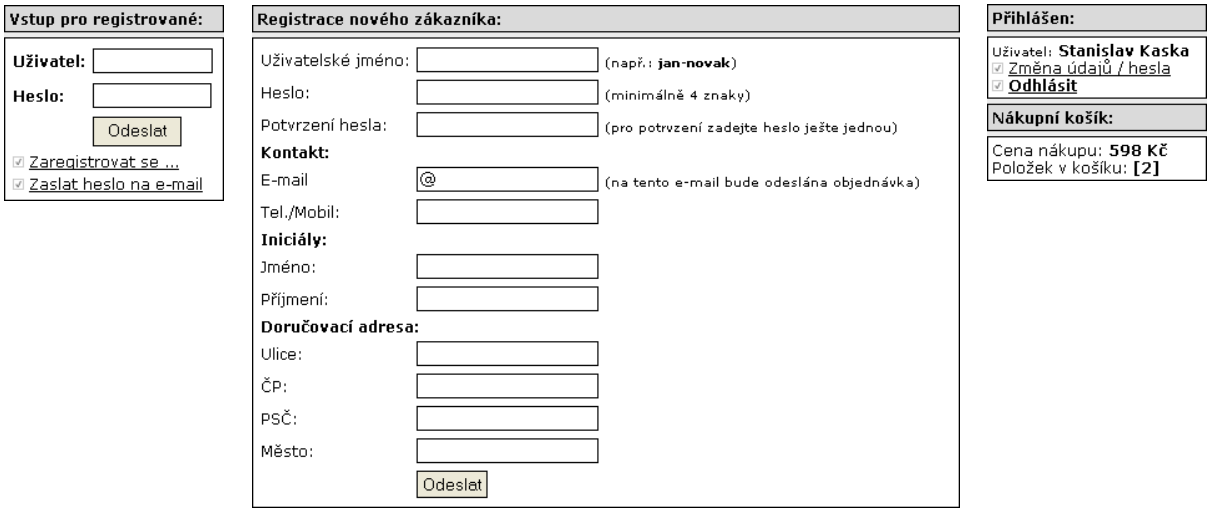
Obr. č. 7.3.1. – Formulář pro preferované produkty

8. Uživatelská sekce

Uživatel internetového obchodu může prohlížet nabízené produkty, vhodit do košíku a proto aby mohl vytvořit objednávku, musí být zaregistrovaný. K vyhledání požadovaného zboží je několik možností. Pomocí stromu kategorií, výrobců v levé části obchodu a vyhledáváni podle výrazu.

8.1. Registrace

Zákazník internetového obchodu, který má v úmyslu si nabízené zboží objednat je povinen vyplnit registraci a to svědomitě a pravdivě. Kliknutím na odkaz zaregistrovat, se zobrazí formulář, ve kterém se vyplní požadované údaje ke zpracování objednávky. Registrační údaje obdrží na uvedený email. Po úspěšné registraci se přihlásí ve formuláři pro registrované uživatele. Po přihlášení se zobrazí informace o přihlášeném uživateli, kde je možnost změnit kontaktní údaje a heslo. Také je zde odkaz pro odhlášení a stav nákupního košíku. Odhlášeni se provede i při uzavření okna prohlížeče.



Obr. č. 8.1.1. – Registrace a přihlášení uživatele

8.2. Vyhledání zboží a zobrazení detailu

V levé části obchodu je strom, kde si uživatel vybere kategorii nebo podkategorii, o které si myslí, že bude obsahovat hledaný výrobek. Produkty lze setřídit i podle výrobce. Následně se produkt zobrazí s minimálními informacemi a malým náhledem. Pro detailní informace o produktu slouží odkaz **Zobrazit detail**. Dále má uživatel možnost hledat produkt pomocí výrazu, který se zadává do formuláře **Hledat produkt**. Zobrazí se tři seznamy. V prvním jsou výsledky hledání podle názvu produktu, ve druhém jsou výsledky hledáni podle popisu produktu a v posledním seznamu jsou výsledky podle názvu kategorie.

Obr. č. 8.2.1. – Stručný detail o produktu s malým náhledem

Výsledky hledání slova (výrazu): "slon"

Výsledky hledání v **produktech** podle **názvu**.

Název:	Cena:	Detail:	Akce:
Slon modrý	55 Kč	zobrazit	Do košíku
Hračka plyš - Slon - BS-374	362 Kč	zobrazit	Do košíku
Kousátko chladící Slon	58 Kč	zobrazit	Do košíku
Hračka plyš - Kniha Slon - BS-581	337 Kč	zobrazit	Do košíku
Hračka plyš - Hrazdička Slon - BS-354	270 Kč	zobrazit	Do košíku
Bryndák s chrastítkem na ručičku Slon	110 Kč	zobrazit	Do košíku
Hračka plast - Hrazda - PL-3009 - Slon	425 Kč	zobrazit	Do košíku

Výsledky hledání v **produktech** podle **popisu**.

Název:	Cena:	Detail:	Akce:
Hračka plyš - Slon - BS-374	362 Kč	zobrazit	Do košíku
Chrastící bačkorky 3K1701E	169 Kč	zobrazit	Do košíku

Výsledky hledání podle **názvu kategorie**.

Nebyly nalezeny žádné produkty

Obr. č. 8.2.2. – Výsledky hledání

Obr. č. 8.2.3. – Detail produktu

8.3. Objednání zboží

Produkt lze vhodit do košíku a to odkazem **Do košíku**. Než však produkt bude do košíku vhozen, zobrazí se stručné informace o produktu a formulář, v němž je možnost vybrat provedení nebo barvu produktu. V případě, že administrátor uvedl provedení nebo barvu v popisu produktu, musí uživatel položku vyplnit sám. Za názvem provedení nebo barvy je uvedena cena jemu odpovídající. Dále je nutno uvést počet kusů.

Obr. č. 8.3.1. – Před vhozením do košíku

Po odeslání formuláře se zobrazí seznam zboží v košíku, kde uživatel můžeme případně některý z produktů odstranit a to kliknutím na ikonu červeného přeškrtnutí, nebo zobrazit podrobnější informace o některém z produktů a to kliknutím na odkaz s obrázkem oka. Uživatel má možnost tento seznam produktů v košíku kdykoliv zobrazit a to kliknutím na odkaz **Nákupní košík** v horní pravé části obchodu.

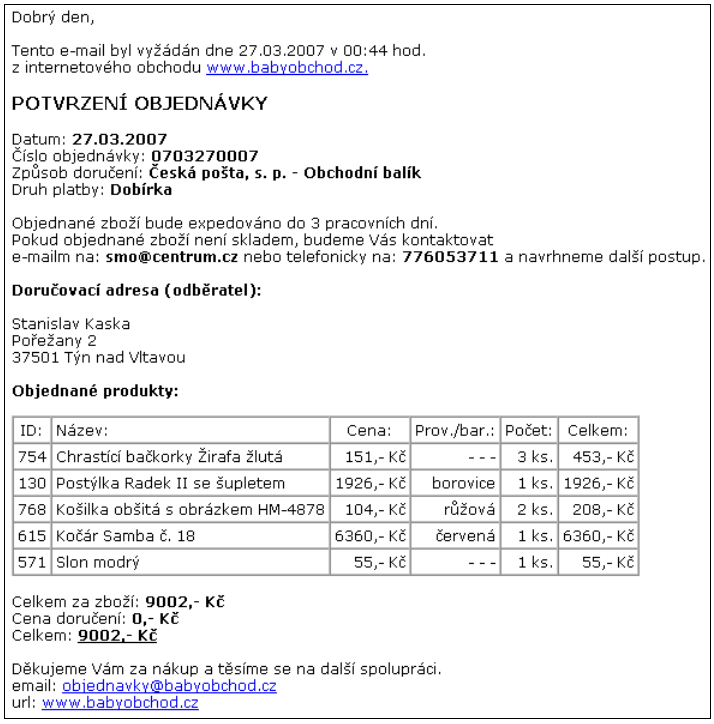
Obsah košíku:

PČ:	Název:	Cena:	Prov.:	Kusů:	Za ks.:	Det.:	Odeb.:
1	Postýlka Radek II se šupletem	1926 Kč	borovice	1 ks.	1926 Kč		
2	Slon modrý	55 Kč	- - -	1 ks.	55 Kč		
3	Kočár Samba č. 18	6360 Kč	červená	1 ks.	6360 Kč		
4	Košilka obšitá s obrázkem HM-4878	104 Kč	růžová	2 ks.	208 Kč		
5	Chrastící bačkorky Žirafa žlutá	151 Kč	- - -	3 ks.	453 Kč		

Celkem cena za zboží: 9002 Kč

Obr. č. 8.3.2. – Obsah košíku

Pokud je uživatel zaregistrován a přihlášen, může objednávku odeslat. Není-li tomu tak, může tak učinit dodatečně. Před samotným odesláním objednávky je třeba vybrat způsob platby. Následně se zobrazí informace o úspěšném zpracování objednávky a v případě, že zákazník uvedl v registraci pravdivý kontaktní email, bude mu doručeno potvrzení objednávky.



Dobrý den,

Tento e-mail byl vyžádán dne 27.03.2007 v 00:44 hod.
z internetového obchodu www.babyobchod.cz.

POTVRZENÍ OBJEDNÁVKY

Datum: **27.03.2007**
Číslo objednávky: **0703270007**
Způsob doručení: **Česká pošta, s. p. - Obchodní balík**
Druh platby: **Dobírka**

Objednané zboží bude expedováno do 3 pracovních dní.
Pokud objednané zboží není skladem, budeme Vás kontaktovat
e-mailm na: **smo@centrum.cz** nebo telefonicky na: **776053711** a navrhneme další postup.

Doručovací adresa (odběratel):

Stanislav Kaska
Pořežany 2
37501 Týn nad Vltavou

Objednané produkty:

ID:	Název:	Cena:	Prov./bar.:	Počet:	Celkem:
754	Chrastící bačkorky Žirafa žlutá	151,- Kč	- - -	3 ks.	453,- Kč
130	Postýlka Radek II se šupletem	1926,- Kč	borovice	1 ks.	1926,- Kč
768	Košilka obšitá s obrázkem HM-4878	104,- Kč	růžová	2 ks.	208,- Kč
615	Kočár Samba č. 18	6360,- Kč	červená	1 ks.	6360,- Kč
571	Slon modrý	55,- Kč	- - -	1 ks.	55,- Kč

Celkem za zboží: **9002,- Kč**
Cena doručení: **0,- Kč**
Celkem: **9002,- Kč**

Děkujeme Vám za nákup a těsíme se na další spolupráci.
email: objednavky@babyobchod.cz
url: www.babyobchod.cz

Obr. č. 8.3.3. – Potvrzení objednávky

9. Zdrojové kódy

Celý internetový obchod je vytvářen pomocí funkcí, které jsou všechny v jednom souboru a to `funkce.php` , který je před volanými soubory obchodu začleňován pomocí příkazu `include 'funkce.php';` , aby bylo možno tyto funkce využívat. Jedná se o soubory `index.php` , `kosik.php, detail.php` a další soubory potřebné pro administraci.

9.1. Struktura MySQL databáze

Všechny data se vkládají do tabulek v databázi. Proto, aby byl obchod funkční, je třeba aby byli tyto tabulky vytvořené. Jedná se o tabulky `kategorie` , `kosik` , `pref_prod` , `produkty` , `uzivatele` , `vyrobci` a `zobrazeni` . Pro vytvoření těchto tabulek slouží tyto dotazy:

```
-- Struktura tabulky `kategorie`
CREATE TABLE `kategorie` (
  `id` smallint(6) NOT NULL auto_increment,
  `id_kat` tinyint(4) NOT NULL default '0',
  `id_podkat` tinyint(4) NOT NULL default '0',
  `nazev` varchar(27) collate cp1250_czech_cs NOT NULL,
  PRIMARY KEY  (`id`),
  FULLTEXT KEY `nazev` (`nazev`)
) ENGINE=MyISAM DEFAULT CHARSET=cp1250 COLLATE=cp1250_czech_cs PACK_KEYS=0;
-- Struktura tabulky `kosik`
CREATE TABLE `kosik` (
  `id` smallint(6) NOT NULL auto_increment,
  `id_uziv` smallint(6) NOT NULL default '0',
  `id_prod` smallint(6) NOT NULL default '0',
  `prov_bar` varchar(17) collate cp1250_czech_cs default NULL,
  `pocet` smallint(6) NOT NULL default '0',
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=cp1250 COLLATE=cp1250_czech_cs PACK_KEYS=0;
-- Struktura tabulky `pref_prod`
CREATE TABLE `pref_prod` (
  `id` smallint(6) NOT NULL default '0',
  `produkt` varchar(17) collate cp1250_czech_cs NOT NULL default '',
  `vyhody` varchar(50) collate cp1250_czech_cs NOT NULL default '',
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=cp1250 COLLATE=cp1250_czech_cs;
-- Struktura tabulky `produkty`
CREATE TABLE `produkty` (
  `id` smallint(6) NOT NULL auto_increment,
  `id_kat` smallint(6) NOT NULL default '0',
  `id_podkat` smallint(6) NOT NULL default '0',
  `id_vyrobce` smallint(6) NOT NULL default '0',
  `nazev` varchar(50) collate cp1250_czech_cs NOT NULL default '',
  `cena` tinytext collate cp1250_czech_cs NOT NULL,
  `prov_bar` tinytext collate cp1250_czech_cs NOT NULL,
  `popis` text collate cp1250_czech_cs NOT NULL,
  `prodano` int(11) NOT NULL default '0',
  `obrazek` tinytext collate cp1250_czech_cs NOT NULL,
  `poznamka` varchar(30) collate cp1250_czech_cs NOT NULL default '',
  PRIMARY KEY  (`id`),
  FULLTEXT KEY `nazev` (`nazev`),
  FULLTEXT KEY `popis` (`popis`)
) ENGINE=MyISAM DEFAULT CHARSET=cp1250 COLLATE=cp1250_czech_cs PACK_KEYS=0;
-- Struktura tabulky `uzivatele`
CREATE TABLE `uzivatele` (
  `id` smallint(6) NOT NULL auto_increment,
  `uzivatel` tinytext collate cp1250_czech_cs NOT NULL,
  `heslo` tinytext collate cp1250_czech_cs NOT NULL,
  `email` tinytext collate cp1250_czech_cs NOT NULL,
  `mobil` tinytext collate cp1250_czech_cs NOT NULL,
  `jmeno` tinytext collate cp1250_czech_cs NOT NULL,
  `prijmeni` tinytext collate cp1250_czech_cs NOT NULL,
  `ulice` tinytext collate cp1250_czech_cs NOT NULL,
  `cp` smallint(6) NOT NULL default '0',
  `psc` tinytext collate cp1250_czech_cs NOT NULL,
  `mesto` tinytext collate cp1250_czech_cs NOT NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=cp1250 COLLATE=cp1250_czech_cs PACK_KEYS=1;
-- Struktura tabulky `vyrobci`
CREATE TABLE `vyrobci` (
  `id` smallint(6) NOT NULL auto_increment,
  `nazev` varchar(20) collate cp1250_czech_cs NOT NULL default '',
  `obrazek` varchar(20) collate cp1250_czech_cs NOT NULL default '',
  PRIMARY KEY  (`id`),
  FULLTEXT KEY `nazev` (`nazev`)
) ENGINE=MyISAM DEFAULT CHARSET=cp1250 COLLATE=cp1250_czech_cs PACK_KEYS=0;
-- Struktura tabulky `zobrazeni`
CREATE TABLE `zobrazeni` (
  `id` smallint(6) NOT NULL auto_increment,
  `id_kat` smallint(6) NOT NULL default '0',
  `id_podkat` smallint(6) NOT NULL default '0',
  `id_vyr` smallint(6) NOT NULL default '0',
  `id_prod` smallint(6) NOT NULL default '0',
  `pocet` mediumint(9) NOT NULL default '0',
  `nazev` varchar(50) collate cp1250_czech_cs NOT NULL default '',
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=cp1250 COLLATE=cp1250_czech_cs PACK_KEYS=0;
```

9.2. Soubor `funkce.php`

Tento soubor má velikost 117 kB, obsahuje 2290 řádků s 89 funkcemi. Jsou v něm obsaženy všechny funkce potřebné k funkčnosti obchodu. V následující části je popsáno několik nejpoužívanějších nebo zajímavých funkcí.

9.2.1. Zpracování dotazu do databáze `zpracuj_databazi($dotaz)`

Tato funkce je nejčastěji volána. Vrací výsledky z databáze zpracované dotazem, který je jako vstupní parametr. Funkce se volá `$vysledky = zpracuj_databazi($dotaz);` , kde v proměnné `$vysledky` jsou požadovaná data.

```
function zpracuj_databazi($dotaz) {
 require 'konfigurak.php';
 $spojeni = mysql_connect ($server, $uzivatel, $heslo)
            or die ("Nepovedlo se připojit k databázi!");
 mysql_select_db($databaze, $spojeni)
                 or die ("Nepovedlo se otevřít databázi! ".mysql_error());
 mysql_query("set names cp1250", $spojeni);
 $vysledky = mysql_query($dotaz, $spojeni)
             or exit ("Špatný dotaz, dotaz nelze provést! ".mysql_error());
 mysql_close($spojeni);
 return $vysledky;
}
```

V počátku je zaváděn soubor `konfigurak.php` , ve kterém jsou definovány proměnné potřebné pro navázání správného spojení s MySQL serverem. Výhoda je v tom, že po přenesení obchodu na jiný server, stačí změnit tyto údaje a obchod bude funkční.

```
$server = "127.0.0.1";
$uzivatel = "root";
$heslo = "";
$databaze = "babyobchodcz";
```

9.2.2. Identifikace uživatele `nastavit_session($uzivatel, $heslo)`

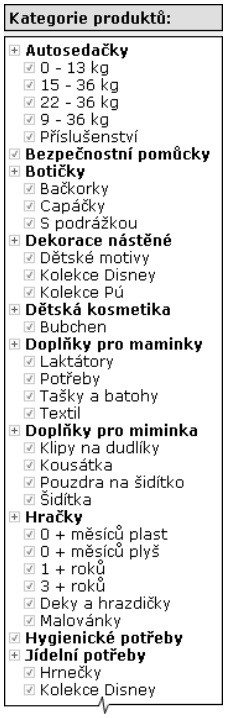
Pro správný chod obchodu je třeba zjistit o jakého se jedná uživatele, je-li přihlášen atd. To se zajistit tím, že nastavíme proměnnou `$_SESSION['id_uziv']`, která se ukládá na server, je kdykoliv dostupná. Zaniká při zavření prohlížeče. Do této proměnné se uloží jednoznačný identifikátor reprezentující daného uživatele. Pro aktivaci je třeba provést příkaz `session_start();` .

```
function nastavit_session($uzivatel, $heslo) {
 if (($uzivatel != "") && ($heslo != "")) {
  if (overit_uzivatele($uzivatel, $heslo) == true ) {
   $_SESSION['id_uziv'] = cislo_uzivatele($uzivatel, $heslo);
   $_SESSION['spatne_reg_udaje'] = "";
   if ($_SESSION['id_n_uziv'] != "") {
    redef_kosiku($_SESSION['id_n_uziv'], $_SESSION['id_uziv']); }
   $_SESSION['id_n_uziv'] = "";
  }
  else { $_SESSION['spatne_reg_udaje'] = "ano"; }
 }
 else {
  if (($_SESSION['id_n_uziv'] == "") && ($_SESSION['id_uziv'] == "")) {
   $_SESSION['id_n_uziv'] = cislo_nezareg_uzivatele(); }
  $_SESSION['spatne_reg_udaje'] = "";
 }
}
```

Vstupní parametry do této funkce je proměnná `$uzivatel` , obsahující hodnotu o uživatelském jménu a proměnná `$heslo` , obsahující heslo uživatele, zakódované ve formátu MD5, zadané ve formuláři pro přihlášení. Tyto proměnné se testují, byli-li zadány. Pokud ano, zavolá se funkce `overit_uzivatele()` , pro ověření těchto údajů v databázi. Souhlasí-li tyto údaje, volá se funkce `cislo_uzivatele()` , která vrací jednoznačný identifikátor o uživateli, který se vkládá do proměnné `$_SESSION['id_uziv']` . Musí se také počítat s tím, že než se uživatel přihlásil, může mít v košíku vhozené zboží. V podstatě by po přihlášení v košíku bylo žádné zboží, protože by byl obchodem vidět jako jiný uživatel. Proto se testuje proměnná `$_SESSION['id_n_uziv']` , kde je jednoznačná informace o neregistrovaném uživateli. Volá se funkce `redef_kosiku()` , která všechno zboží v košíku vložené jako neregistrovaného uživatele předefinuje na registrovaného uživatele. V poslední fázi se vytváří funkcí `cislo_nezareg_uzivatele()` jednoznačný identifikátor pro neregistrovaného uživatele.

9.2.3. Strom kategorie produktů a výrobců `menu()` a `menu_vyr()`

Funkce `menu()` vytiskne všechny názvy kategorií a jejich podkategorii a funkce `menu_vyr()` výrobce z databáze. Tyto názvy jsou tisknuté jako odkazy a tak se stávají základní navigací pro obchod. Odkaz který vypadá například takto: `index.php?&id_kat=2` , se předá informace skriptu `index.php` .



Obr. č. 9.2.3.1. – Kategorie produktů

9.2.4. Funkce `tisk_nejprodavenajsi()` , `tisk_nejzobrazovanejsi()` a `tisk_nejnovejsi()`

Další z nejčastěji volaných funkcí jsou tyto. Zobrazí osm nejprodávanějších, nejzobrazovanějších a naposledy vložených produktů. Při odeslání objednávky obchod zpracuje potvrzení objednávky, které se odešle zákazníkovi na uvedený email a také vedení firmě na email `objednavky@babyobchod.cz` , aby bylo možné objednávku zpracovat. Podle objednaného zboží se změní také hodnota `prodáno` v databázi u daného produktu. Této hodnoty využívá funkce `tisk_nejprodavenajsi()` , která vytiskne osm produktů se stručnými informacemi.

Při každém zobrazení detailu o produktu, zobrazení kategorie, podkategorie nebo zobrazení výrobce se tato akce zaznamenává do databáze. Funkce `tisk_nejzobrazovanejsi()` využívá tyto data a vytiskne tak osm nejvíce zobrazovaných produktů. S těmito daty o počtu zobrazení jednotlivých produktů, kategorií a výrobců je ještě nakládáno jinak. Jedou měsíčně se volá skript, který tyto data zpracuje a vytiskne statistiku o nejvíce zobrazovaných produktech atd. Následně jsou data smazána a začíná se znamenávat od začátku.

Vložením produktu do databáze je přiřazeno jednoznačné číslo ID. Funkce `tisk_nejnovejsi()` vytiskne posledních osm vložených produktů podle ID.

9.2.5. Převzorkování obrázku `prevzorkovat($nazev_souboru, $nazev_souboru_logo)`

První vstupní parametr funkce je název obrázku z adresáře `zaloha_obr` , který byl do tohoto adresáře nahrán pomocí funkce `nahrani_souboru_uloz_udaje()` . Druhý vstupní parametr je název obrázku reprezentující logo výrobce, umístěný v adresáři `zaloha_obr/loga` . Funkcí `getimagesize()` lze získat rozměry obrázku. S těmito rozměry se dále pracuje, tak aby maximální velikost byla 550 x 550 pixelů a přitom nebyl obrázek nijak zdeformován. Proměnnou `$zmenseny` se reprezentuje nový přerozměrovaný obrázek a proměnnou `$zdroj` původní obrázek. Funkcí `imagecopyresampled()` se do přerozměrovaného obrázku vloží obrázek původní. Dále se funkcí `magefilledrectangle()` vloží bíle vyplněný čtyřúhelník daných rozměrů, který se orámuje černě pomocí funkce `imagerectangle()` . Do tohoto rámečku se vloží text `www.babyobchod.cz` černé barvy. V případě, že název loga výrobce není prázdný řetězec, zjistí se rozměry obrázku. Proměnná `$zdroj_logo` reprezentuje tento obrázek. Funkcí `imagecopymerge()` se sloučí logo s přerozměrovaným obrázkem. Následně se logo orámuje černě a uloží funkcí `imagejpeg()` do adresáře `obr` pod stejným názvem jako původní obrázek. Výsledný obrázek je znázorněn dále.

```
function prevzorkovat($nazev_souboru, $nazev_souboru_logo) {
 $soubor = "zaloha_obr/".$nazev_souboru;
 list ($sirka, $vyska) = getimagesize($soubor);
 if ($sirka > $vyska)
  { $nova_sirka = 550; $nova_vyska = $vyska * (550 / $sirka); }
 else { $nova_vyska = 550; $nova_sirka = $sirka * (550 / $vyska); }
 $zmenseny = imagecreatetruecolor($nova_sirka, $nova_vyska);
 $zdroj = imagecreatefromjpeg($soubor);
 imagecopyresampled($zmenseny, $zdroj, 0, 0, 0, 0,
                    $nova_sirka, $nova_vyska, $sirka, $vyska);
 $cerna_barva = imagecolorallocate($zmenseny, 0, 0, 0);
 $bila_barva = imagecolorallocate($zmenseny, 255, 255, 255);
 imagefilledrectangle($zmenseny, 3, ($nova_vyska - 21), 129,
                      ($nova_vyska - 3), $bila_barva);
 imagerectangle($zmenseny, 3, ($nova_vyska - 21), 129,
                ($nova_vyska - 3), $cerna_barva);
 imagestring($zmenseny, 3, 7, ($nova_vyska - 19),
             "www.babyobchod.cz", $cerna_barva);
 if ($nazev_souboru_logo != "") {
  $soubor_logo = "zaloha_obr/loga/".$nazev_souboru_logo;
  list ($sirka_logo, $vyska_logo) = getimagesize($soubor_logo);
  $zdroj_logo = imagecreatefromjpeg($soubor_logo);
  imagecopymerge($zmenseny, $zdroj_logo, ($nova_sirka - $sirka_logo - 3),
                 ($nova_vyska - $vyska_logo - 3), 0, 0, $sirka_logo,
                  $vyska_logo, 100);
  imagerectangle($zmenseny, ($nova_sirka - $sirka_logo - 3),
                 ($nova_vyska - $vyska_logo - 3), ($nova_sirka - 3),
                 ($nova_vyska - 3), $cerna_barva);
 }
 imagejpeg($zmenseny, "obr/".$nazev_souboru);
}
```

Obr. č. 9.2.5.1. – Výsledný přerozměrovaný obrázek s logem

Podobná funkce `prevzorkovat_na_maly($nazev_souboru)` převzorkuje obrázek na maximální rozměry 177 x 177 pixelů, který se použije jako malý náhled.

9.2.6. Vyhledání produktu podle výrazu `hledej($hled_vyraz)`

K hledání výrazu v databázi funkce využívá speciální dotaz, který vypadá obecně takto:

```
select * from nazev_tabulky where match (nazev_sloupce)
                            against ('hledany_vyraz')
```

Sloupec `nazev_sloupce` v tabulce `nazev_tabulky` musí být označen jako `fulltext` .

V obchodu je použito více dotazů s různými modifikacemi, pro efektivnější nalezení hledaného výrazu. Je zde například použit operátor hvězdička `*` s příkazem `in boolean mode` a to například takto:

```
$dotaz[2] = "select * from produkty where match (".$sloupec[$i].")
             against ('".$hled_vyraz."*' in boolean mode) limit 0,20";
```

Operátor hvězdička `*` říká, že za tímto výrazem mohou následovat znaky, nehledá se tak pouze hledaný výraz jako slovo, ale slova začínající na tento výraz.

9.3. Soubor `index.php`

Soubor `index.php` je XHTML dokument se začleněnými PHP skripty. Je základním souborem reagujícího na akce uživatele a podle toho vykonává funkce. Reaguje na vstupní proměnné reprezentující kategorii, podkategorii i výrobce, aktuální stránku atd. Hodnoty jsou předávány pomocí odkazu, tím pádem metodou GET a ve skriptu se pracuje tedy s polem `$_GET[]`. Předání odkazem vypadá takto: `index.php?&id_kat=1&id_podkat=1&id_vyr=&strana=2&od=18`. Za názvem souboru se uvede otazník `?` , za nímž se dále uvádějí jednotlivé proměnná a jejich hodnoty. Proměnná začíná znakem `&` , kde za tímto znakem se uvede název klíče pro pole `$_GET[]`. Dále znak rovná se = a odpovídající hodnota. Takto se může pokračovat dále.

```php
<?php
 include 'funkce.php';
 $cas_zacatek = mikro_cas();
 session_start();
 nastavit_session($_POST["uzivatel"], $_POST["heslo"]);
 nastavit_bar_schema($_GET["bar_schema"]);
 if ($_GET["odhlasit"] == "ano") { odhlasit_uzivatele($_SESSION["id_uziv"]); }
?>
```

```php
<?php zmena_bar_schema($_SESSION["bar_schema"]) ?>
```

```php
<?php
 kde($_GET["id_kat"], $_GET["id_podkat"], $_GET["id_vyr"], $_GET["strana"]);
?>
```

```php
<?php horni_menu(); ?>
```

```php
<?php
 hledat(); menu(); menu_vyr(); pocitadlo_toplist();
 statistika_zobrazení($_GET["id_kat"], $_GET["id_podkat"], $_GET["id_vyr"], "");
?>
```

```php
<?php
 if ($_POST["hledej"] != "")
  { echo ("Výsledky hledání slova (výrazu): \"".$_POST["hledej"]."\""); }
 else {
  echo ("Nacházíte se v kategorii: ");
  kde($_GET["id_kat"], $_GET["id_podkat"], $_GET["id_vyr"], $_GET["strana"]);
}
?>
```

```php
<?php
 if ($_POST["hledej"] != "") { hledej($_POST["hledej"]); }
 else {
  if (($_GET["id_kat"] == "") && ($_GET["id_podkat"] == "") &&
      ($_GET["id_vyr"] == "") && ($_GET["od"] == "") &&
      ($_GET["strana"] == "")) {
   tisk_top(); tisk_top_text();
   strankovani($_GET["id_kat"], $_GET["id_podkat"],
               $_GET["id_vyr"], $_GET["strana"]); }
  else {
   strankovani($_GET["id_kat"], $_GET["id_podkat"],
               $_GET["id_vyr"], $_GET["strana"]);
   tisk($_GET["id_kat"], $_GET["id_podkat"], $_GET["id_vyr"], $_GET["od"]);
   strankovani($_GET["id_kat"], $_GET["id_podkat"],
               $_GET["id_vyr"], $_GET["strana"]);
  }
 }
?>
```

```php
<?php
 uziv_prihlasen($_SESSION["id_uziv"]);
 tisk_nejprodavenajsi(); tisk_nejzobrazovanejsi();
 tisk_nejnovejsi(); bar_schema();
?>
```

```php
<?php
 podpis();
 $cas_konec = mikro_cas();
 $cas = $cas_konec - $cas_zacatek;
 echo "<small>Tato stránka byla vygenerována za: "
      .round($cas, 3)." sekund.</small>";
?>
```

9.4. Soubor `kosik.php`

Zpracovává nákupní košík obchodu. Hodnoty do souboru vstupující jsou opět předávány odkazem a to v podobě `kosik.php?&objednat=571` nebo `kosik.php?&odebrat=ano&id=21` . Tento soubor je XHTML dokument s vnořenými PHP skripty. Úvodní a konečná část PHP skriptu je stejná jako u `index.php`. Tiskne se stejné úvodní menu, navigační strom kategorii a výrobců a nejprodávanější, nejzobrazovanější a nejnovější produkty. Tělo vypadá takto:

```
<?php
 redef_kosiku($_SESSION["id_uziv"], $_SESSION["id_n_uziv"]);
 if ($_GET["objednat"] != "") { tisk_pred_kosikem($_GET["objednat"]); }
 if (($_POST["id"] != "") && ($_POST["pocet"] != "")) {
  if ($_SESSION["id_uziv"] != "") {
   vloz_do_kosiku($_SESSION["id_uziv"], $_POST["id"],
                  $_POST["prov_bar"], $_POST["pocet"]); }
  else {
   vloz_do_kosiku($_SESSION["id_n_uziv"], $_POST["id"],
                  $_POST["prov_bar"], $_POST["pocet"]); }
 }
 if ($_GET["odebrat"] == "ano") { odebrat_zkosiku($_GET["id"]); }
 if ($_POST["objednat"] == "ano") {
  vystavit_objednavku($_SESSION["id_uziv"], $_POST["druh_pladby"] );
  prodan_produkt($_SESSION["id_uziv"]);
  vyprazdnit_kosik($_SESSION["id_uziv"]);
 }
?>
```

```
<?php
 if ($_SESSION["id_uziv"] != "") { tisk_kosiku($_SESSION["id_uziv"]); }
 else { tisk_kosiku($_SESSION["id_n_uziv"]); }
?>
```

```
<?php
 if ($_POST["objednat"] == "ano") { vyrizeno(); }
 else {
  if (polozek_vkosiku($_SESSION["id_uziv"], $_SESSION["id_n_uziv"]) >= 1) {
   info_zakaznik($_SESSION["id_uziv"]); }
  else { nic(); }
 }
?>
```

9.5. Soubor `detail.php`

Tento soubor je volaný pomocí JavaScriptu a to proto, aby se zobrazil v novém okně prohlížeče podle uvedených parametrů. Odkaz na tento soubor vypadá takto:

```
<a href="detail.php?&id=607"
   onclick="window.open(this.href, '_blank','width=757,height=713,
            menubar=no,status=no,scrollbars=yes'); return false">
 Zobrazit detail
</a>
```

Atributem `href` se předává do souboru hodnota `id`, aby bylo zřejmé, jakému produktu se má tisknout detail. Atribut `onclick` využívá události myši po kliknutí. Volá JavaScript, který metodou `windows.open()` otevírá nové okno prohlížeče. Parametry této metody říkají, co se zobrazí v okně, jaké rozměry bude mít okno, zobrazí-li se menu prohlížeče nebo rolovací lišta atd.

V souboru `detail.php` je funkce, která zajistí kompletní zobrazení daného produktu se všemi informacemi a obrázky s názvem `tisk_detail($_GET['id'], $_GET['obr_vel'])`.

10. Úprava obrázků

Většinu obrázků k produktům bylo nutné nafotit a upravit tak, aby se daly vhodně použít v obchodu. Určitou část upravování při vkládání obrázku k produktům zajišťují skripty, které automaticky při vložení změní rozměry obrázku převzorkováním, vloží logo výrobce a text o tom, že obrázek byl vytvořen obchodem www.babyobchod.cz. Po nafocení produktu je nutné obrázek "ořezat", tak aby bylo pozadí - okolí obrázku bílé a vynikal pouze samotný produkt. To je nutné provádět manuálně.

Zdrojový obrázek z digitálního fotoaparátu je nafocen zpravidla v rozlišení 1600 x 1200 pixelů. Velikost souboru bývá v rozmezí 500 – 800 kB ve formátu JPG. Tento obrázek je zbytečně velký pro prezentaci v obchodu, ale dostačující pro práci v grafickém programu pro vyříznutí produktu. Výsledný, již ořezaný obrázek s rozměry kolem 1000 x 1000 pixelů, ve formátu JPG, s velikostí v rozmezí 100 – 200 kB, je již vhodný pro vložení k produktu. Tento obrázek se při vkládání zálohuje na server a dále upravuje již podle popsaného algoritmu. Úpravu obrázku provádím v programu Macromedia Fireworks MX 2004.

Obr. č. 10.1. – Zdrojový obrázek z digitálního fotoaparátu a výsledný ořezaný obrázek

11. Závěr

Myslím si, že uživatelé, spotřebitelé začali více důvěřovat internetových obchodům a začínají směle objednávat zboží, neboť vědí, že jim bude včas a v požadované kvalitě doručeno třeba až z druhého konce republiky domů. I díky rozšiřujícímu se připojení k internetu v domácnostech, roste nákup zboží v internetových obchodech. V budoucnu bude dosti pravděpodobné, že většina zboží pro domácnost bude objednávána pomocí internetu. Právě k tomuto účelu slouží internetový obchod, který jsem vytvořil. Máte možnost jej navštívit na adrese: `http://www.babyobchod.cz` .

12. Použitá literatura

[1] Petr Pexa, Jazyky XHTML, DHTML, CSS a WML, Kopp České Budějovice 2006

[2] Dušan Janovský, HTML, CSS, JavaScript, `http://www.jakpsatweb.cz`

[3] Adam Jun, MySQL manuál, `http://mm.gene.cz`

[4] PHP Documentation Group, PHP Manuál, `http://www.php.net/docs.php`

[5] MySQL Manuál, `http://www.mysql.com/documentation/`