1. Úvodem

Současná moderní společnost klade stále vyšší nároky na všechny úseky a obory našeho života. Stále více se ukazuje, že mít rychlé a kvalitní informace bývá pro danou oblast rozhodující. To platí zejména pro podnikatelskou činnost, má-li být úspěšná a na patřičné úrovni. Právě této oblasti věnuji pozornost ve své diplomní práci.

Absolutně nezanedbatelnou službou je INTERNET. Jako nejvíce výhodné se mi jeví používání internetových aplikací klient /sever (případně používání aplikací tohoto typu). Jedním z důvodů je používání výkonného centrálního počítače, který je schopen velice rychle vyhledat námi požadované informace. Dalším důvodem je, že jeho struktura a uspořádání zamezuje zbytečnému přenosu dat a tím snižuje celkové zatížení sítě. Výhodou služby na Internetu je skutečnost, že může být i klasickou komerční službou. Tak se do prostředí Internetu přenesou některé obchody, instituce, případně soutěže.

Předkládanou diplomovou prací chci splnit povinnost nejen vůči škole a svému studiu na této škole, ale jako úspěch bych považoval její využití v praktickém podnikatelském životě.

1.1. Jak se zmocnit nového nástroje řízení, marketingu, komunikace, reklamy?

Základním předpokladem pro správné rozhodnutí o využívání Internetu jako nového nástroje řízení, marketingu, komunikace a reklamy je mít dostatek fundovaných informací. Přestože je Internet počítačová síť, ty nejdůležitější informace nemají s počítačovou terminologií nic společného. Jde totiž o informace, jakým způsobem bude Internet ovlivňovat naše podnikání na přelomu tisíciletí.

V nejbližší budoucnosti bude totiž Internet ovlivňovat podniky v rovině strategické, řídící, v rovině marketingu a komunikace i v rovině nových produktů a služeb. Podstatou problému tedy je, jak se manažeři s tímto dnes již nezvratným

faktem vypořádají. K Internetu se je třeba postavit s plnou vážností, jako ke každému jinému problému. Základem řešení je tedy definování cílů, kterých má být dosaženo, analýza možných cest postupu k těmto cílům a výběr optimální varianty. Co to může konkrétně znamenat v praxi?

Odborníci upozorňují, že Internet je schopen prostoupit celý organizmus podniku. To znamená, že se může dotknout nejen vrcholových manažerů, ale i ekonomů, personalistů, účetních, skladníků a pod. Z toho vyplývá úkol vypracovat základní analýzu stávajících cílů a analýzu možností Internetu pro každý konkrétní úsek a vyhodnotit možné přínosy pro tento úsek. V posledním období se ukázalo, že podnik, který neměl s oblastí komunikace či výpočetní techniky nic společného, se vydal cestou internetového byznysu, protože přijal Internet jako výzvu, příležitost k novému způsobu podnikání, ve kterém se karty na trhu teprve rozdávají a všichni zájemci jsou seřazeni na startovaní čáře.

Internet umožňuje komunikovat každému marketingovému pracovníkovi s potenciálním klientem, získávat od něj bezprostřední reakce na prezentovanou nabídku a získávat řadu dalších cenných informací. Také obchodníci (možná i živnostníci) by si mohli najít v Internetu zalíbení. Mohou jeho prostřednictvím nabídnout svým stávajícím i předpokládaným zákazníkům aktuální informace jak o cenách, o stavu svých zásob, nových výrobcích i o poskytovaných službách. Těm odvážnějším pak možná přijde na mysl obrátit dosavadní průběh zakázek v podniku (obchodě či živnosti) a místo přijímání objednávek začít aktivně zásobovat své odběratele. To předpokládá, že odběratel vystaví aktuální stav položek dodávaného zboží a dodavatel se stará, aby tento stav neklesl pod smluvenou hranici. Princip takové změny spočívá v tom, že prodejce prodává a nestará se o nákup, protože je producentem průběžně zásobován.

Podnik, který se rozhodl pro zavedení a využití Internetu, by měl mít svůj internetový projekt. Je zapotřebí střízlivě odhadnout možnosti technické, personální, finanční i časové tak, aby budování Internetu v podniku nezastavilo chod úseků, kterých se tato zásadní změna týká. Projekt by se měl rozhodně zabývat strukturou obsahu prezentace, případně strukturou internetu. Je to jedna

z nejdůležitějších podmínek úspěchu. Měla by být logická a pro všechny snadno pochopitelná.

Jedním z ožehavých problémů je **aktualizace údajů po Internetu**. Do prezentace by mělo zasahovat co nejméně lidí, a proto je vhodné v projektu vyřešit bezpečnostní politiku Internetu. V podstatě jde o schéma, které stanovuje cestu informace od jejího vzniku až k jejímu schválení a vystavení. Naopak v případě zpětné vazby cestu od získání informace až po její vyhodnocení, případně jak na ni v podniku reagovat.

Internetová reklama by jako součást projektu měla být výsledkem vlastního průzkumu Internetu. Je vhodné najít na Internetu místa, kam podnik (firma) umístí reklamní odkaz na svoji prezentaci.

Informace by měly být aktuální, proto je potřebné tyto často ověřovat. V odborné literatuře se uvádí nejméně jedenkrát za měsíc. Internet je fenomén, který hýbe světem. Často ale vyvolává rozporné reakce. Jedno je jisté. Ukázal se být příliš dobrým obchodem na to, aby jej ty největší společnosti pustily z rukou. Navíc objektivně přináší stále komfortnější služby těm, kteří ho využívají. Bohužel pravdou je, že zatím na tom není Česká republika po stránce telekomunikační nejlépe, ale technické problémy jsou řešitelné a postup Internetu nezastaví. Zpomalit ho může jen myšlení lidí a nepochopení jeho praktického využití.

1.2. Využití programu jazyka JavaScript pro prezentaci firmy a formulář na objednávku zboží

Ke splnění úkolu, který je mi dán tématem diplomní práce se mi jeví použití programu jazyka JavaScript jako nejvhodnější. Tento jazyk umožňuje jak prezentaci firmy, tak i vypracování formuláře na objednávku zboží.

2. Jak pracuje JavaScript.

Na standardním Web serveru se dostane více informaci tím, že se klikne na hyperodkaz a server zašle jiný soubor. Na interaktivnějších stránkách se vyplní formulář, odešle na server a čeká se na odpověď. V každém případě se musí čekat až server pošle nový soubor. Touto informací je většinou nová stránka, která může mít jakoukoliv podobu. Na stránkách rozšířených pomocí JavaScriptu jsou do HTML vloženy kódy JavaScriptu. JavaScript může tuto informaci nabídnout okamžitě, bez nutnosti čekání na server, nebo na připojení k Internetu. Tato informace může pocházet z uživatelského vstupu, kódu v HTML dokumentu, nebo v jiných rámcích, nebo jiných oknech. Tato rozšířená stránka zviditelní tuto novou informaci aktualizací obsahu formuláře, nebo vygenerováním celého nového dokumentu.

2.1. JavaScript a HTML dokument.

JavaScript pracuje s prohlížeči tak, že kód vloží přímo do stránky HTML. Firma Netscape vyvinula nový obecný tag s názvem SCRIPT, aby prohlížeč rozpoznal skriptovací jazyk. Chceme-li informovat prohlížeč, že kód je JavaScript, musí se do tagu SCRIPT přidat atribut LANGUAGE = „JavaScript“ (jestliže toto schází předpokládá prohlížeč, že jde o JavaScript). Mnoho kódů je v JavaScriptu těmito tagy uzavřeno tak, jak ukazuje následující příklad:

```
 < SCRIPT LANGUAGE = „JavaScript“>
a = „Nazdar!“
//Uložení do poměnné a hodnotu „nazdar“
< /SCRIPT>
```

Mezi tagy SCRIPT můžeme psát dva typy kódu: přímé **příkazy** a **funkce**. Přímý příkaz prohlížeč vykoná po načtení. Objekty jsou například inicializovány přímými příkazy. Funkce jsou bloky kódů, které jsou vykonány jiným kódem,

nebo událostí. Kliknutí myší jako událost například obvykle spustí funkci. Mnoho existujících tagů HTML má nyní dodatečné atributy pro podporu JavaScript.

Všechny prvky formuláře mohou být nyní například označeny prvkem NAME . Použití atributu NAME k identifikaci objektů v dokumentech zjednoduší kódy a ladění programů. Posledním vylepšením HTML je rozpoznání událostí (například kliknutí myší, změny v textových panelech a načtení, nebo uvolnění stránky z paměti). Jinými slovy způsob, jakým dokument rozpozná uživatelovu akci. Tyto události se používají pro spuštění akce JavaScriptu.

Například:

```
<FOFM>

<P> Klikněte uvnitř a mimo

<INPUT TYPE = "TEXT" NAME ="TEXT" onChange ="sample"
value="a">

</FORM>
```

Kód JavaScriptu je spuštěn událostí, může být jednoduchý nebo složený. U jednoduché akce může být celý kód vložen do prvku události. Tento případ je na předcházejícím příkladu v sample. value =“a“

2.2. Syntaxe JavaScriptu

JavaScript je založen na akčně orientovaném modelu Word Wide Webu. Prvky Web stránky (tlačítky, zátržítky,...) mohou spouštět akce, , nebo události. Jestliže nastane jeden z těchto případů, vykoná se odpovídající část kódu JavaScriptu, což je obvykle funkce JavaScriptu. Stiskneme-li například tlačítko SUBMIT v online formuláři objednávky, může se vyvolat funkce JavaScript, která prohlédne obsah formuláře a zkontroluje, zda uživatel zadal všechny požadované informace.

2.3. Proměnné

Jedna z největších odlišností mezi JavaScriptem a většinou ostatních jazyků spočívá v tom, že JavaScript nemá explicitní datové typy. Není zde možné určit, zda určitá proměnná představuje číslo (integer, real) , nebo řetězec (string). Všechny proměnné mohou být inicializovány tak, že při jejich deklaraci je jim přiřazena hodnota, nebo mohou zůstat neinicializovány. Kvůli čitelnosti kódu je výhodné proměnné inicializovat příkazem. Například:

```
var x = 7
var s = "Text"
var p = null
```

2.3.1. Numerické hodnoty

V JavaScriptu existují dva numerické typy: celá čísla a desetinná čísla. Celá čísla mohou být určena ve formátu se základem desítkovém, osmičkovém, nebo hexadecimálním. Jednotlivé tvary jsou rozlišovány podle prvního nebo druhého znaku následovně:

´ 1- 9 následovaná jakýmkoliv číslem je desítkové číslo

´ 0 následovaná souborem číslic 0 - 7 je osmičkové číslo

´´0x nebo 0X následované 0 - 9 nebo A - F je hexadecimální číslo

2.3.2. Řetězce

Řetězce v JavaScriptu mohou být uvedeny buď v jednoduchých uvozovkách (‘Text’), nebo v dvojitých uvozovkách (“Text“). Jestliže se řetězec začne s jedním typem uvozovek, musí se také tímto typem ukončit.

2.4. Početní operátory

+	sčítání, spojování řetězců
-	odčítání, unární negace
*	násobení
/	dělení
%	modulo
++	preinkrement, postinkrement
- -	predekrement, postdekrement

2.4.1. Logické operátory

= = , ! =	rovnost, nerovnost
< ,< = , = > , >	aritmetické a řetězové srovnání
!	logické NOT
& & , II	logické AND, logické OR
?	podmínkový výběr
,	logické spojení

2.4.2. Bitové operátory

& , I	bitová AND, bitová OR
^	bitové a exkluzivní OR (XOR)
~	bitové not
< < , > > , > > >	posun do leva, posun doprava, posun doprava bez znaménka

2.4.3. Operátory přiřazení

=	přiřazení

2.5. Řídící struktury

2.5.1. Příkaz if

Příkaz if se používá k podmíněnému vykonání jednoho bloku kódu. Má dva tvary. Samostatný if a příkaz if..else. Samostatný příkaz if je shodný s testovaným výrazem, známým jako if test, a z blok kódu, který je proveden, pokud je výraz vyhodnocen jako booleovská hodnota true.

Příklad:

```
if (test) {
příkaz
}else {
            příkaz pro else
}
```

2.5.2. Příkaz while

Příkaz while se používá k vykonávání bloku kódu v případě, že je určitá podmínka true. Formát tohoto příkazu je následující:

```
while (test ) {
příkaz
}
```

Podmínka test je vyhodnocena jako logický výraz. Jestliže je true, vykonají se příkazy bloku v závorkách. Potom se vrátíme cyklem zpět na začátek příkazu while a podmínka test je vyhodnocena znovu. Tento proces pokračuje tak dlouho dokud test je false, nebo nějaký příkaz uvnitř bloku cyklus nuceně přeruší. Například příkazem break.

2.5.3. Příkaz for.

Tento příkaz patří mezi nejvýkonnější a nejsložitější ze tří sazeb, které ovlivňují chod kódu. Primárním účelem příkazu je opakování bloku příkazů pro určitý rozsah hodnot. Příkaz for má následující tvar:

```
for ( počpřík; testpřík; aktpřík; ) {

}
```

Podmínka for má tři části, které jsou odděleny středníky. Počpřík se obvykle používá k inicializaci proměnné, i když bychom mohli použít na tomto místě jakýkoliv jiný příkaz (počpřík se vykoná pouze jednou pří prvním průchodu příkazem for).

Testpřík je podmínkový test a slouží k přesně stejnému účelu jako v příkazu while. Je testován na začátku každého cyklu. Příkaz for je ukončen, jestliže bylo vyhodnocena podmínka jako false.

Aktpřík je vykonán na konci každého cyklu, jako by byl umístěn bezprostředně za posledním příkazem v bloku for. Obvykle se používá k aktualizaci proměnné, která je inicializována částí počpřík.

2.5.4. Příkaz with

Cílem příkazu with je umožnit vytvoření v jeho objektu (nebo instanci) několik odkazů na objekt, bez nutnosti opakovat název tohoto objektu.

Formát příkazu:

```
with (názevobj) {
příkaz
}
```

názevobj je název objektu nebo instance. Uvnitř bloku with se vyskytují odkazy na vlastnosti objektu názevobj, jako by před nimi byl objekt názevobj a tečkový operátor (.).

2.6. Funkce a objekty

Funkce a objekty představují nejvyšší úroveň organizace v rámci jazyka JavaScript.

2.6.1. Funkce

Funkce je blok kódu, který má název. Kdykoliv je toto jméno použito, je funkce vyvolána, čili je vykonán kód v rámci funkce. Funkce mohou být také vyvolány s hodnotami, kterým se říká parametry funkce. Parametry se mohou použít uvnitř samotné funkce. Funkce slouží ke dvěma účelům:

1) jsou organizačním nástrojem, což znamená, že nám dovolují vykonávat stejné operace bez nutnosti kopírovat tejný kód
2) druhý účel funkcí JavaScriptu je propojení akcí na Web stránce s kódem JavaScript

Funkce JavaScriptu včetně vhodných tagů ve zdrojovém HTML dokumentu mohou být vyvolány například tím, že uživatel klikne myší, stiskne tlačítko, vybere text, či provede jinou akci.

Syntaxe zápisů funkce v JavaScriptu je následující:

```
Function Název (seznam parametrů) {
telo funkce
}
```

2.6.2. Objekty

Objekty slouží ke stejnému účelu jako funkce, ale pracují s daty. Až od toho okamžiku jsme datové položky chápali jako jednoduché proměnné, deklarované slůvkem var. Každá z těchto beztypových kvalit může v daném okamžiku obsahovat pouze jednu hodnotu určitého druhu. Objekty mohou obsahovat více hodnot. To znamená, že skupinu příbuzných textových prvků můžeme sloučit do jedné. To co se v JavaScriptu nazývá objekty, je ve většině ostatních jazyků nazý-

váno strukturou dat (nebo třídou). Jedním z mnoha objektů je také objekt document. Výstup do okna Web prohlížeče provádíme například document.write ('TEXT').

2.7. Události v JavaScriptu

Když přejedete myší přes hypertextový odkaz, vytvoří se událost mouseover, když provedete načtení nebo uvolnění stránky z paměti jedná se o další dvě události. Pomocí tohoto je možné vytvářet různé interakce uživatele s uživatelem.

Události prvky HTML	Blur	Click	Change	Focus	Load	Mouseover	Select	Submit	Unload
Button		X							
Ceckbox		X			X				X
Document								X	
Form						X			
Link		X							
Radio		X							
Reset		X							
Selection	X		X	X					
Submit		X							
Text	X		X	X			X		
Textarea	X		X	X			X		

JavaScriptu používá pro manipulaci s textem model, který definuje čtyři události, přiřazené k textovým panelům a polím, ale nikoliv textovým panelům pro zadávání hesla - change, select, focus a blur. Událost change (změna) se generuje při každé změně textu a událost select (výběr) se generuje při každém vybrání textu. Vybrat text neznamená pouze kliknout v editačním textovém panelu nebo poli. Je nutné skutečně vybrat, nebo označit část textu pomocí myši. Událost focus (zaměření) a blur (zaostření) jsou o něco složitější. Říkáme, že textový panel, nebo pole je *zaměřeno*, pokud aktivně přijímá vstup z klávesnice. Textová položka je zaměřena, jestliže je kliknuto kdekoliv uvnitř textu. Událost blur je opakem události focus. Zaostření se objeví, jakmile již položka není zaměřena. To se stane v případě, když je zaměřena jiná položka, nebo se zaměření ztratilo.

Příklad:

```
<HTML>
<SCRIPT LANGUAGE="JavaScript">
<!--
function dontclickme(){
  alert ("Tlačítko bylo stisknuto")
  return(false)
}
-->
</script>
<BODY>
<FORM>
<INPUT TYPE="button" NAME="mycheck" value="HA!" onC-
lick="dontclickme()">
</FORM>
</BODY>
</HTML>
```

2.7.1. Použití handlerů událostí.

Události JavaScriptu se vyskytují na třech úrovních.

- na úrovni celého WWW dokumentu.
- na úrovni individuální části <FORM>
- na úrovni prvku části <FORM>

Handlery událostí submit v tagu FORM

Tag FORM se používá na začátku definice formuláře HTML. Obsahuje atributy METHOD používané k potvrzení formuláře a ACTION používané

k převzetí. Může také obsahovat další jednotlivý typ atributu handleru události, atribut onSubmit.

```
 <FORM NAME="nazevFormulare"... onSUB-
MIT="submithandler()">
```

Handler onSubmit je vyvolán, když má být potvrzen obsah formuláře. Jedná se o vrcholnou akci, týkající se celého formuláře. Používá se pro ověřování platnosti formuláře.

2.7.2. Handlery událostí v prvcích formuláře

Téměř všechny prvky formuláře mohou mít jeden nebo více handlerů událostí. Obecně řečeno, pomocí tlačítek lze vytvořit události click, zatímco textové a výběrové položky mohou vytvořit události focus, blur, select a change.

2.8. Objekty v JavaScriptu

2.8.1. Vestavěné objekty

Mezi vestavěné objekty patří řetězcové objekty Date a Math. Jako vestavěné se označují proto, že skutečně nemají nic společného s Web stránkami, jazykem HTML,URL, prostředím aktuálního prohlížeče nebo ničím viditelným.

2.8.2. Řetězcové objekty

Každá proměnná, jejíž hodnotou je řetězec, je ve skutečnosti řetězcový objekt. Při vytváření řetězcových objektů se nepoužívá slovo new. Řetězcové objekty mají jednu vlastnost, length, a mnoho metod. Vlastnost length udává délku řetězce. Metody se dělí na tři skupiny:

1. metody, které pracují s obsahem řetězce
2. metody, které pracují se vzhledem řetězce
3. metody, které převádějí řetězec na prvek HTML.

2.8.3. Metody pracující s obsahem řetězců

Metody toLovewerCase a toUpperCase převádí celý obsah řetězce buď na malá, nebo velká písmena.

Metody charAt a substring se používají při výpisu buď jednoho znaku z řetězce (který je na pozici idx), nebo řady znaků (od pozice fromidx do pozice toidx). Pozice znaků začíná od nuly, stejně jako pole JavaScriptu, takže všechny indexy musí být mezi 0 a číslem o jednu menším než je velikost pole.

Metody indexOf i lastIndexOf se používají při vyhledávání znaku (chr) v řetězci. Metoda indexOf vyhledává od začátku (zleva) řetězce a metoda lastIndexOf vyhledává od konce řetězce (zprava). V případě že najdou znak, obě navrací celočíselný index. V případě, že znak nenajdou, navrátí -1.

2.8.4. Metody vzhledu řetězce.

Používají se při úpravě výsledného vzhledu řetězce na Web stránce. Metody vzhledu řetězce umožňují získat stejný vzhled v JavaScriptu bez nutnosti použití odpovídajících prvků HTML.

Jsou to:

```
big()
blink()
bold()
fixed()
fontcolor(color)
fontsize()
italics()
small()
strike()
sub()
```

2.8.5. Metody řetězců HTML

JavaScript nabízí dvě metody pro konverzi řetězců na hyperodkazy. Tyto metody se používají při vytváření zdrojových textů v HTML, zatímco objekty HTML už jsou součástí jazyka HTML. Do této kategorie patří dvě metody:

```
anchor(namestr)
link(hrefstr)
```

Obě se používají při vytváření některé formy atributu ukotvení (<A>). Rozdíl mezi nimi je ten, že metoda se používá při vytvoření ukotvení s hodnotou **namestr** atributu NAME, zatímco metoda link se používá při vytvoření ukotvení s atributem HREF nastavením na **hrefstr**. Jinými slovy, metoda anchor vytvoří ukotvení, které je cílem, zatímco metoda link vytvoří ukotvení, které je odkazem. Obě metody konvertují řetězec, se kterým pracují, na textovou část tohoto ukotvení. Hodnotou **namestr** musí být platný řetězec, který smí být atributem MANE, takže by neměl obsahovat žádné vložené mezery. Argument **hrefstr** by měl obsahovat platné URL, protože uživateli bude umožněno, aby na něj kliknul.

2.8.6. Objekt Math

Objekt Math se používá při různých matematických kalkulacích. Má některé vlastnosti, mezi něž patří standardní konstanty, například PI=3,14159... a dále velkou kolekci metod, které představují běžné trigonometrické a algebraické funkce. Všechny metody Math se zabývají čísly s desetinou tečkou. U úhlů se očekává zadání v **radiánech**, nikoliv ve stupních.

Objekt Math má tyto vlastnosti:

```
E
LN10
LN2
PI
SQRT1_2
```

```
SQRT2
Objekt Math má tyto metody:
abs(číslo)
acos(číslo)
asin(číslo)
atan(číslo)
ceil(číslo)
cos(rad)
exp(číslo)
floor(číslo)
min(číslo1,číslo2)
max(číslo1,číslo2)
pow(číslo1,číslo2)
random()
round(číslo)
sin(rad)
sqrt(číslo)
tan(rad)
```

Mezi metody objektu Math patří běžné trigonometrické funkce sinus (sin), kosinus (cos), tangent (tan) a jejich převrácené hodnoty arcsinus (asin), arckosinus (acos), arctangent (atan). Každá z těchto funkcí přijímá úhel v radiánech a jejich výsledkem je desetinné číslo.

Metody ceil,floor a round akceptují při zadávání desetinná čísla, přičemž výstupem je celé číslo. Metoda ceil navrací nejmenší celé číslo, které je větší nebo rovno jejímu argumentu a metoda floor navrací největší celé číslo, které je menší, nebo rovno jejímu argumentu. Metoda round navrací nejnižší celé číslo.

Metody exp, log, pow a sqrt se všechny zabývají umocňováním, nebo odmocňováním. Metoda exp umocňuje vlastnost Mat. E číslem, které je stanoveno jako její argument a je opakem metody log, která vrací přirozený logaritmus svého argumentu,který musí být kladný. Metoda pow umocní číslo1 číslem 2. Metoda sqrt navrací odmocninu svého argumentu.

A nakonec metody abs, min, max a random vykonávají různé užitečné operace. Metoda abs navrací absolutní hodnotu svého argumentu. Metody min a max vrací minimální hodnotu svých dvou argumentů. Metoda random nepoužívá žádné argumenty. Navrátí náhodné desetinné číslo v rozmezí 0 až 1.

2.8.7. Objekt Date

Zpracování dat a času je jedna z nejhorších úloh v každém jazyce. Je to proto, že mnoho lidí upřednostňuje udávání dat a časů v nedesítkových systémech. Měsíců je 12, hodin 24, minut a vteřin je 60. Všechny tyto varianty jsou z pohledu počítače poměrně nelogické. Rád pracuje s pěknými, zaokrouhlenými čísly, preferuje mocniny 2 a násobky 10. Aktuálně připomínám problém v počítačovém světě při přechodu z roku 1999 na rok 2000. Zatím se celosvětově hledá schůdné řešení.

Objekt Date zjednodušuje a automatizuje mnoho operací spojených s převodem dat z podoby čitelné pro člověka do interní podoby čitelné pro počítač. Objekt Date JavaScriptu využívá standardu z UNIXu, používaný při interním uchovávání data a času jako počet milisekund od 1.ledna 1970. Toto datum se často označuje jako epocha , protože krátce předtím byl systém UNIX poprvé představen počítačovému světu.

Objekt Date nemá žádné vlastnosti ,ale mnoho metod. Aby bylo možné používat objekt Date, je nutné vytvořit základní instanci Date:

```
New Date()
New Date(datestring)
New Date(yr,mon,day)
```

První způsob vytvoří instanci Date, která představuje aktuální datum a čas. Druhý způsob používá řetězec ve formě „Měsíc Den, Rok“ například „November 23, 1997“ a převede ho na instanci Date. Tento řetězec může na konci volitelně obsahovat čas ve tvaru HH:MM:SS, který se používá při nastavení času (HH-hodiny, MM-minuty, SS-vteřiny). Hodiny by měly být zadány pomocí 24hodinového formátu.Třetí způsob používá tři celá čísla představující rok, měsíc a den (**měsíc je vždy číslován od nuly**).

Objekt Date má velké množství metod pro získání metod pro získání a nastavení komponent data. Jsou to:

```
getDate()
getDay()
getHours()
getMinutes()
getSecond()
getTime()
getTimeZoneOffset()
getYear()
setDate()
setHours()
setMinutes()
setMonth()
setSeconds()
setTime()
setYear()
```

Většina těchto metod vykonává operace zřejmé už z jejich názvu. Metoda getMonth() vrací číslo určující měsíc a to v rozmezí od 0 do 10. Metoda getTime vrací interní podobu data v JavaScriptu, tj. počet milisekund od Epochy. Tato po-

slední metoda se může zdát pochybná, ale je užitečná při srovnání dvou dat, které z nich je starší. Metody set se používají při nastavení různých součástí instance Date.

Kromě metod get a set má objekt Date také metody pro převod instance Date na řetězec a dvě statické pro analýzu dat. Tyto metody jsou:

```
toGMTString()
toLocateString()
toString()
parse()
UTC()
```

První tři metody převodu instancí Date na řetězec představující datum a čas vzhledem ke greenwichskému střednímu času (GMT, někdy označovaný také jako UTC-univerzální čas), vzhledem k aktuálnímu formátu data (který je například odlišný v Evropě a v USA) jako obyčejný řetězec. Poslední dvě metody se používají při převodu řetězce data v místním času (metoda parse), nebo univerzálním času na počet milisekund od Epochy. Na tyto metody se musí používat odkazy, jako na metody Date.parse() a date.UTC(), protože jsou statické. Z toho důvodu se nemohou používat s instancemi objektu Date. Protože navrací interní podobu data, jsou tyto hodnoty často upraveny pomocí metody setTime.

2.9. Vestavěné funkce

Zatímco objekt Date pracuje jako skutečný objekt s výjimkou svých dvou statických metod, objekt String je téměř neviditelný. Všechny normální programy v JavaScriptu pracují s řetězci tak, jakoby se jednalo o samostatný datový typ.

V samotném JavaScriptu je vestavěna také malá skupina funkcí. Nejsou to metody a nikdy se nepoužívají s instancí, která používá tečkový operátor. Jsou na stejné úrovni jako, funkce které vytvoříte pomocí slova funcion. V současnosti existuje pět vestavěných funkcí. Jsou to:

```
escape(str)
```

```
eval(str)
parseFloat(str)
parseInt(str,radix)
unEscape(str)
```

Funkce escape a unEscape se používají při převodu na, nebo z konvence únikového kódu, kterou používá HTML. Takže funkce escape(“ “) vrátí hodnotu %20.

Vestavěná funkce parseFloat se pokouší převést její řetězcový argument na desetinné číslo. Převádění řetězce str pokračuje tak dlouho, dokud funkce nenarazí na znak, který nemůže být částí platného desetinného čísla,.

Funkce eval vyhodnotí svůj řetězcový argument jako výraz JavaScriptu a nahradí jeho hodnotou. Všechna normální pravidla pro vyhodnocování výrazů se provádějí pomocí funkce eval. Tato funkce vyhodnocuje jakýkoliv výraz JavaScriptu, bez ohledu co znamená.

Příklad:

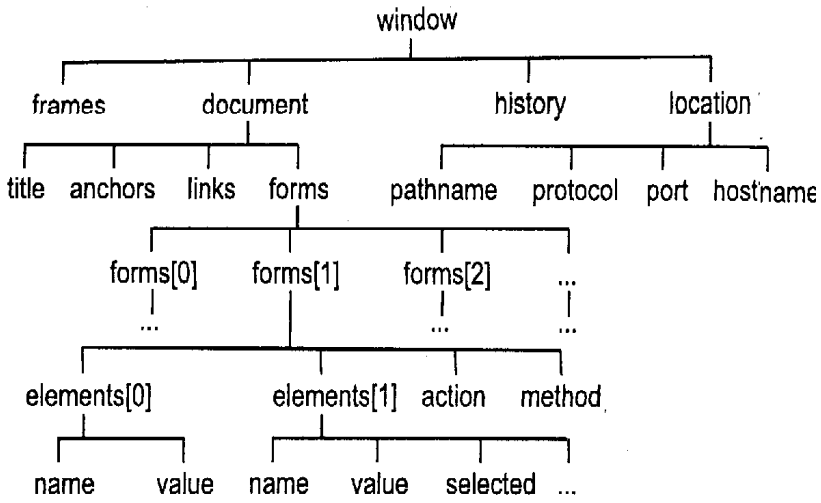
```
x = 10
z = eval(“(x * 14)-(x/2) + 11“)
```

2.10. Objekty prohlížeče a jazyka HTML

Objekty JavaScriptu představují velmi zajímavou skupinu vestavěných objektů, metod a funkcí, které nabízí vše, co je očekáváno od moderního programovacího jazyka. Umožňují řídit strukturu, zapouzdření, funkce automatické operace atd. protože je JavaScript navržen pro práci s WWW, existují také propojení mezi tímto jazykem a obsahem stránek HTML.

Toto propojení nabízí v JavaScriptu bohatá sada objektů prohlížeče a jazyka HTML. Objekty prohlížeče odráží prostředí prohlížeče (objekty, které se mohou použít při odkazu na aktuální stránku, seznam historie a aktuální URL). Existují také metody pro otevření nového okna, vyvolání dialogových panelů a přímé psaní kódu HTML (document.write).

Objekty prohlížeče jsou na vrcholu hierarchie objektů JavaScriptu, protože představují souhrnné informace a akce, které jsou nutně spojené s určitou WWW stránkou. Na každé WWW stránce má každý prvek HTML odpovídající objekt v hierarchii objektů. Konkrétně každý formulář HTML a každý prvek HTML ve formuláři má odpovídající objekt. Na obrázku je možno vidět přehled hierarchie objektů JavaScriptu.



2.10.1.Objekty prohlížeče

Hlavními objekty prohlížeče podle pořadí důležitosti jsou následující:

- window
- frames
- document
- location
- history

2.10.2.Objekt window

Jak vyplývá z obrázku 1, objekt window (okno) je nejvyšší objekt v hierarchii objektů JavaScriptu. Každé okno prohlížeče, které je právě otevřené, bude mít odpovídající objekt window. Všechny ostatní objekty jsou následníky objektů window. Každé okno je spojeno s danou Web stránkou a struktura této stránky se odráží v objektu docunent tohoto okna. Každé okno odpovídá nějakému URL, toto URL se odráží v objektu location. Každé okno má historii předchozích stránek zobrazených v tomto okně, které představují různé vlastnosti objektu history.

JavaScript udržuje myšlenku aktuálního okna, takže téměř žádné odkazy na podobjekty aktuálního okna se na něj nemusí odkazovat. Takže lze použít document.write() místo windows.document.write(). Objekty window mají tyto metody:

```
alert(msgstr)

close()

confirm(msgstr)

open(urlstr,wname)

prompt(msgstr)
```

Metody alert a confirmse používají pro zobrazení argumentu msgstr v dialogovém panelu. Metoda alrt se používá při zobrazení něčeho, s čím nemůže uživatel nic udělat. Výstražný dialogový panel obsahuje pouze jedno tlačítko OK. Dialogový panel confirm zobrazí svou zprávu jak s tlačítkem OK, tak i Storno. Jestliže uživatel stiskne na tlačítko OK, pak metoda confirm vrátí hodnotu true, jinak vrací hodnotu false. Metoda prompt se používá při žádosti o uživatelský vstup ve formě řetězce. Metoda zobrazí dialogový panel s řetězcem msgstr a s editačním textovým panelem. Tato metoda také akceptuje druhý volitelný argument, který se použije pro nastavení implicitní hodnoty do textového panelu. Tato metoda vrací to, co zadal uživatel, ve formě řetězce. Pro otevření nového okna prohlížeče, je určena metoda open. Argument urlstr je řetězec, představující URL, jež bude otevřeno v okně. Argument wname je řetězec, který udává název

nového okna. Tato metoda vrací instanci objektu window (která značí, že bylo vytvořeno nové okno). Zároveň akceptuje také třetí parametr, který například určuje zda se má zobrazit panel nástrojů. Jakmile je vyvolána metoda close z instance windows, vlastní okno se uzavře a jeho URL se uvolní z paměti.

2.10.3.Objekt document

Každé okno je spojeno s objektem document. Objekt document obsahuje vlastnosti pro každé ukotvení, odkaz a formulář na stránce a také všechny podprvky těchto prvků. Obsahuje také vlastnosti pro atributy stránky title (obsah pole <TITLE> na stránce - záhlaví), barvu popředí (vlastnost fgColor), barvu pozadí (bgColor), různé barvy odkazů a další atributy samotné stránky. Objekt document má tyto metody:

```
clear()
close()
open()
write()
writeln()
```

Metoda clear se používá při úplném smazání okna dokumentu. Je smazán celý obsah okna bez ohledu na to, jak se tam tyto informace dostaly. Metoda open a close se používají při spuštění a ukončení výstupu z bufferu. Metoda write se používá při vypsání řetězce, obsahujícího tag HTML, do aktuálního dokumentu. V případě, že je zadán více než jeden argument, každý z argumentů je chápán jako řetězec a vypíše se. Metoda writeln je stejná jako metoda write, pouze po ukončení výpisu jejího argumentu (nebo argumentů) se odřádkuje. Toto odřádkování bude ovšem v jazyce HTML ignorováno s vyjímkou případu, když je metoda writeln uvedena v předformátovaném textu (pomocí tagů <PRE>...</PRE>).

2.10.4.Objekty history a location

Objekty history (historie) se používá při odkazu na seznam už navštívených URL. Objekt history má vlastnost známou jako length, která udává počet URL právě obsažených v seznamu historie. Objekt history má také tyto metody:

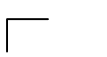
```
back()
forward()
go(where)
```

Metoda go se používá při navigaci seznamem historie. Argument where může být číslo nebo řetězec. Jestliže je argument where číslo, pak toto číslo určuje, jak daleko se má pohybovat v seznamu historie. Kladné číslo znamená, že se chceme pohybovat v tomto seznamu dopředu, zatím co záporné číslo se používá při pohybu zpět. Jestliže argument where obsahuje řetězec představující URL, pak se toto URL načte a stane se aktuálním dokumentem.

Objekt location (adresa) popisuje URL dokumentu. Má vlastnosti, které reprezentují různé části URL. Tyto vlastnosti jsou bohužel často null, proto je vhodné použít metody toString().

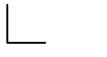
2.10.5.Objekty HTML

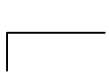
Objekt form odpovídá objektu forms a má podobjekty pro každý ze mnoha prvků formuláře. Nejlépe je to vidět na příkladu, který znázorní ukotvení odkazů, formuláře a prvků formuláře, které jsou v JavaScriptu vedeny jako objekty.



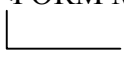
<TITLE>Velmi jednoduchá stránka HTML</TITLE>

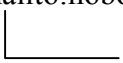
<A NAME="top">Toto je horní část stránky</A>

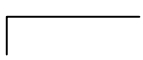




<FORM METHOD="post" ACTION="mailto:nobody@dev.null">







<INPUT TYPE="reset" NAME="já" SIZE=70>

<INPUT TYPE="RESET" VALUE="Storno">

docu-

<INPUT TYPE="submit" VALUE="Ok">

document.forms[0].elements[2].value

Kliknutim se dostanete na

docu-

<A HREF="#top">horní část</A>

docu-

2.11. Rámce a JavaScript

Rámce jsou jeden z nejdůležitějších prvků, které lze přidávat k HTML dokumentům. Umožňují otevření mnohonásobných podoken, nebo panelů na jedné Web stránce. To nabízí zobrazit v jeden okamžik na stejné stránce několik URL. Tím pádem je možno udržovat část obrazovky konstantní, zatím co ostatní stránky jsou aktualizovány.

2.11.1. Specifikace rámců v HTML

Rámce jsou v jazyce HTML organizovány do sad rámců. Jeden z nejdůležitějších a nejvíce matoucích aspektů je vztah rodič potomek mezi rámci, sadami rámců a okny, která je obsahují. První sada rámců, umístěná do okna, má toto okno jako svého rodiče. Sada rámců může také hostit jinou sadu rámců a v tomto případě je výchozí rodičovská sada rámců. Nejvýše umístěný rámec není pojmenován, ale jeho potomci pojmenováni být mohou. Na rámce se lze odkazovat názvy nebo, indexy pole rámců. Skutečný vzhled okna je možné rozdělit příkazem HTML <FRAMESET COLS=40%,*>. Tento příkaz sady rámců rozdělí okno horizontálně na dva rámce. Do rámce nalevo je možno pak přistupovat pomocí frameset[0], do rámce napravo pomocí frameset[1] a nebo je možné použít

také jejich jméno. Okno je také možno rozdělit vertikálně příkazem <FRAMESET ROWS=20%,*,10%>. Tímto způsobem je možné rámce do sebe vnořovat, čím ale vznikají spletité konstrukce kódu.

3. Vytvoření prezentace firmy a formuláře pro objednávku zboží.

Tato práce byla rozdělena do několika dokumentů (souborů). Prvním je **index.html** se zavádí důležité funkce, proměnné, pole, která budou přístupná ze všech dalších dokumentů pomocí direktivy *parent* která toto umožní. Dalšími dokumenty jsou **b.htm** ve kterém jsou nadefinována tlačítka menu, **nabidka.htm** ve které jsou odkazy na jednotlivé druhy zboží, **nakup.htm** v tomto dokumentu je formulář pro objednávku zboží. Dále tato práce obsahuje také dva CGI skripty. První z nich se stará o vytvoření ceníku vybraného druhu zboží, druhý zajišťuje odeslání objednávky zboží. Poslední částí jsou vlastní databázové soubory ceníků.

3.1. Co se stane po zavedení HTML dokumentu?

Po zvedení HTML dokumentu ze souboru index.html se nejprve inicializují pole **nakup** a **zbozi**.

Pomocí příkazu:

```
var nakup = new makeArray(10,4)
var zbozi = new makeArray(40,4)
```

Kde **nakup** je proměnná ve které je uloženo zboží, které si zákazník dal do nákupního koše. V proměnné **zbozi** se ukládají informace o nabízeném zboží. Protože JavaScript nemá vlastní metodu pro vytvoření pole, je nutné vytvořit funkci makeArray, která toto pole vytvoří. Dále je nutné inicializovat proměnnou celkem, ve které je uložen počet položek nabízeného zboží. Nakonec se inicializuje pole adresa, do které se uloží jméno a adresa zákazníka.

Výpis Funkce určené k vytváření pole

```
function makeArray (x,y){
  var count
  this.length = elem
  for (count = 1;count <=elem; count++) this[count] =
new mArray(y);
  return (this)
}
function mArray(elem){
  var count
  this.lenght = elem
  for (count - 1;count <= elem; count++) this[count]
= 0
  return (this)
}
```

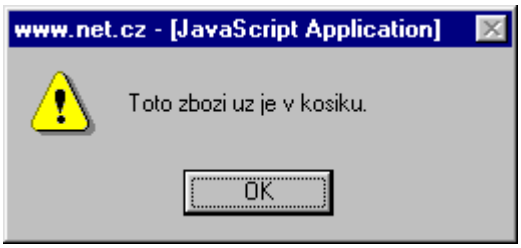
Součástí této funkce je také funkce mArray() která vytváří jednorozměrné pole. Pomocí cyklu For, který zajistí opakování je možno vytvořit pole dvou rozměrné (funkce makeArray). Dále je nutné vytvořit **funkci přidej,** která umožní přidání nového zboží do košíku. Tuto funkci jsem nazval pridej(). Zároveň tato metoda provádí kontrolu zda se v košíku přidávané zboží nenachází.

Výpis Funkce která zajišťuje přidání zboží do košíku

```
function pridej (cislo){
 var je=false
 if (celkem>=2){
    for (count = 1;count <= celkem-1;count++){
      if (nakup[count][1]==zbozi[cislo][1]) {
      alert('Toto zbozi uz je v kosiku.')
```

```
      je=true


      }
    }
  }
if (je==false){
 if (celkem <= 20) {
   nakup[celkem][1] = zbozi[cislo][1]
   nakup[celkem][2] = zbozi[cislo][2]
   nakup[celkem][3] = zbozi[cislo][3]
   nakup[celkem][4] = 1
   celkem++}
 else alert("Nakupni kosik je plny");
  }
}
```

Po spuštění dostane tato funkce jako parametr **cislo** indexové číslo zboží, které si zákazník vybral a chce ho přidat do košíku. Aby nedocházelo k tomu, že požadované zboží bude v košíku několikrát je nutné zjistit jestli se v košíku toto zboží již nenachází. V případě, že v košíku něco je spustí se test na přítomnost tohoto prvku. Toto ověření se provede nejprve zjištěním jestli v košíku vůbec něco je, v případě že ne program provede vlastní přidání zboží. Jestliže je zjištěna přítomnost nějakého zboží v košíku provede se cyklus který, porovná obj. čísla zboží, které chceme přidat s obj. číslem každého zboží které je v košíku. Jestliže jsou nalezena dvě stejná čísla zobrazí program varovné hlášení a funkce se ukončí.

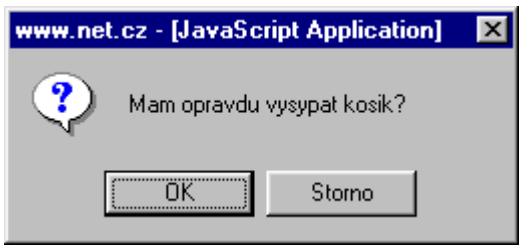


Dále před vložením zboží do košíku je nutné zajistit, aby nedošlo k přetečení maximální indexové hodnoty pole nakup. Přidání zboží se provede:

```
nakup[celkem][1] = zbozi[cislo][1]
nakup[celkem][2] = zbozi[cislo][2]
nakup[celkem][3] = zbozi[cislo][3]
nakup[celkem][4] = 1
```

Každému zboží, při přidání do košíku, je přiřazena hodnota určující množství a nastaví se na hodnotu jedna. Nakonec se zvýší proměnná celkem o jednu.

Poslední metodou v této části dokumentu je VysypatKosik(). Tato metoda provede jen to, že se zeptá jestli to zákazník myslel vážně. V případě, že ano, tak nastaví proměnnou celkem na hodnotu jedna. Tím jakékoliv další přidání začne zapisovat od počátečního indexu.



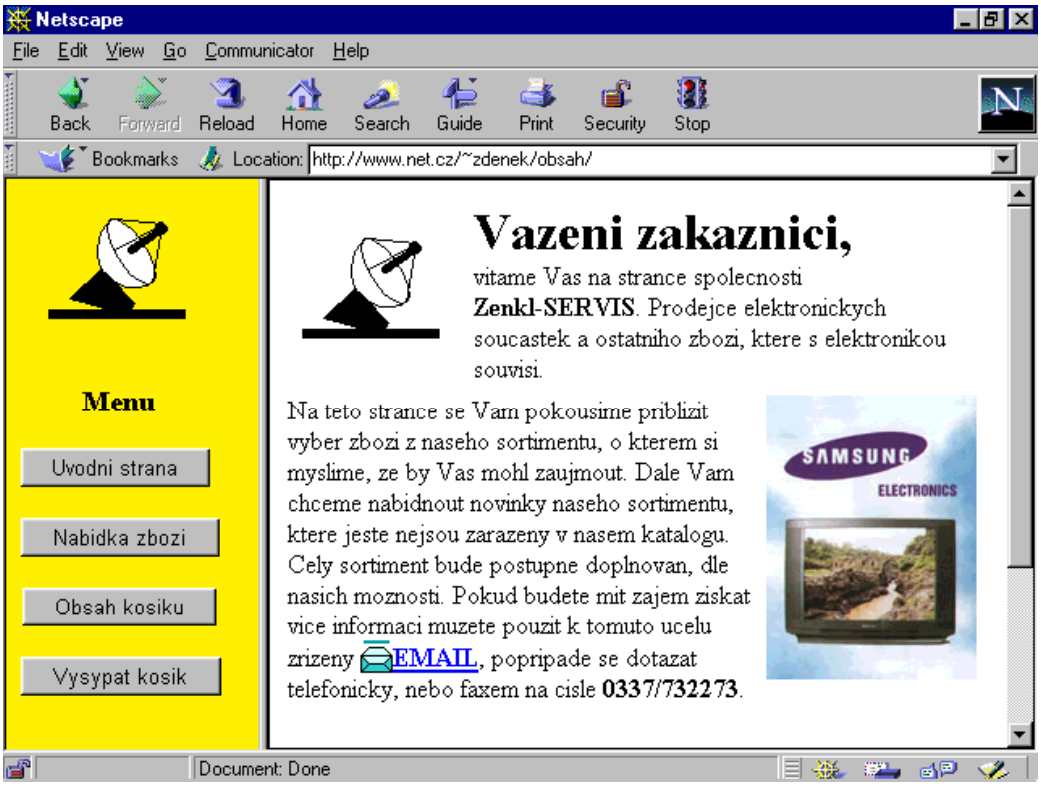
Výpis Funkce VysypatKosik

```
function VysypatKosik(){
 o = confirm('Mam opravdu vysypat kosik?')
   if (o==true){ celkem=1}
}
```

Po zavedení všech těchto nezbytných funkcí proměnných a polí je možno přistoupit dál k vlastnímu rozvržení pracovní plochy dokumentu. Nejprve se provede rozdělení vlastního okna dokumentu na dvě části. Do levé části se vloží uživatelské menu kterým je možno ovládat dokument. Do pravé části se vloží úvodní stránka.Toto se provede pomocí:

```
<FRAMESET cols="160,*">
<FRAME src="b.htm" name="menu" scrol="no">
<FRAME src="uvod.htm" name="obsah">
</FRAMESET>
```

Výsledkem je toto grafické znázornění:

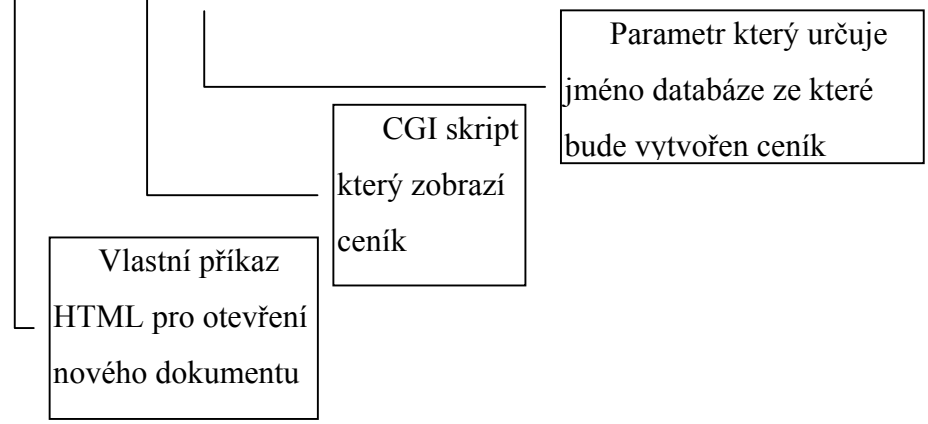


Takže levá strana zůstává statická a pravá strana se mění podle toho co si uživatel zvolil v nabídce. K vlastnímu ceníku se přistupuje tak, že uživatel stiskne tlačítko **Nabídka zboží**.

3.2. Soubor nabidka.htm

Tím se otevře v levé části dokument nabidka.htm, ve které jsou uvedeny jednotlivé druhy zboží. Zboží je rozděleno podle jednotlivých kategorií a druhů. Vlastní ceník se zobrazí poté co uživatel klikne na určitý druh zboží. Vlastní ceník je řešen pomocí CGI skriptu a volá se pomocí HTML (nabidka.htm):

<A HREF="c.cgi?tv">Televizni prijimace</A>



3.3. Soubor C.CGI

Tato část je vypracována pomocí CGI skriptů a generuje vlastní JavaScript. Jelikož JavaScript neumožňuje výstup, ale ani vstup ze souboru. Proto je nutné v této části tohoto dokumentu použít CGI a generovat jím vlastní JavaScript. Nejdříve bych chtěl vysvětlit co to vlastně CGI skript je.CGI (Comon Gateway Interface) je jedna z možností, jak zavést dynamiku do WWW stránek. V zásadě jde o přepis pro program spuštěný na serveru, jak má číst data do WWW serveru a jak má data posílat. Výsledkem je, že při odkazu na link s CGI programem (obvykle nazývaným CGI skriptem), nenatáhne se do WWW klienta tento skript, ale jeho výstup, který je generován tímto programem. To znamená ,že CGI skript zpracuje server a WWW klient dostane jen kód dokumentu, tedy jen výstup z tohoto programu.

Tento program určený pro generování ceníku z databáze je navržen pro programovací jazyk PERL, pro operační systém LINUX.

Po spuštění tohoto skriptu je nejprve nutné zavést hlavičku, která HTML serveru sdělí, že jde o CGI skript a o jaký druh jde. Tímto se zajistí, že výstup z tohoto skriptu bude směrován na WWW klienta. To se provede:

```
#!/usr/bin/perl
print "Content-type: text/html\n\n";
```

Dále je nutné zavést do paměti data z databáze a přepracovat je tak, aby s nimi byl JavaScript schopen pracovat.

```
$file = $ENV{'QUERY_STRING'};
```

Zavede do proměnné $file jméno databázového souboru.

```
open (IN, "cenik/${file}.txt");
 $druh = <IN>;
 @obsah = <IN>;
close(IN);
```

Tato část načte do proměnné $druh celkový název určené skupiny zboží a do proměnné @obsah vlastní obsah databáze.

Je nutné projít každou položku záznamu, zjistit do jaké skupiny jí má zařadit a zajistit vlastní generování JavaScriptu podle obsahu databáze.To se provede tak ,že nejprve se provede kontrola zda se v položkách databáze nenachází nějaký odkaz na obrázek zboží, které je nabízeno. Jestliže ano, je tento název souboru, ve které je uložen, vyříznut a zapsán do proměnné $img. Následně je vygenerována část JavaScriptu, která zajistí, že se obrázek zobrazí v nabídce. Nakonec se provede vytvoření vlastního JavaScriptu, který zapíše do proměnné zboží určené položky. Uvádím kód který to provede.

```
$poc = 0;
```

```
foreach $line (@obsah)
 {
 ++$poc;
 $line =~ s/(.*)(<.*\..*>)(.*)/\1\3/;
 $img = $2;
 if ($img =~ s/<(.*\..*)>/\1/)
  {
   print "parent.zbozi[$poc][4]='$img'\n";
  }
$line =~ s/(.*)#(.*)#(.*)/\1\2\3/;
 print "parent.zbozi[$poc][1]='$1'\n";
 print "parent.zbozi[$poc][2]='$2'\n";
 print "parent.zbozi[$poc][3]='$3'\n";
}
```

Dále se odešle jen kód JavaScriptu který byl vytvořen předem a proto se v závislosti na databázi nemění (až na změnu druhu zboží, který se vypisuje jako nadpis).

Když jsem objasnil vlastní funkci CGI skriptu mohu přistoupit k vlastní funkci JavaScriptu, který je odesílá HTML server a zpracovává ho WWW klient. Ještě připomínám, že hlavní rozdíl mezi JavaScriptem a CGI skriptem je v tom, že JavaScript zpracovává WWW klient, ale CGI skript je zpracován HTML serverem a WWW klient dostává vlastní kód, v tomto případě JavaScript a vlastní HTML.

Po vygenerování tohoto kódu CGI skriptem a načtení do WWW klienta se do proměnné zboží vloží řetězec „není“, toto zajistí vyprázdnění proměnné, která udává jméno souboru ve kterém je uložen obrázek. To se provede pomocí:

```
for (count = 1;count <= 40 count++)
{
```

```
   parent.zbozi[count][4]='neni'
  }
```

Následuje kód JavaScriptu který se mění v závislosti na obsahu určité databáze. Tento kód JavaScriptu má za úkol naplnit proměnnou zboží jednotlivými položkami z databáze a vypadá takto:

```
 parent.zbozi[1][4]='logo.gif'
 parent.zbozi[1][1]='TV01'
 parent.zbozi[1][2]='OTAVA 14 EDC-uhlopricka 37 cm
80pr.'
 parent.zbozi[1][3]='7990'
 parent.zbozi[2][1]='TV02'
 parent.zbozi[2][2]='OTAVA 20 EDC-uhlopricka 51cm
80pr.'
 parent.zbozi[2][3]='9890'
```

Tímto způsobem se do parent.zbozi[x][4] vloží jméno souboru ve kterém je uložen obrázek příslušného zboží. Upozorňuji na to, že tato položka se generuje jen v tom případě kdy záznam obsahuje jméno souboru, ve kterém je obsažen obrázek určeného zboží. Toto zajišťuje zmíněný CGI skript. Do proměnné parent.zbozi[x][1] se zapisuje řetězec, který udává objednávkové číslo zboží. Proměnná parent.zbozi[x][2] obsahuje bližší informace o daném zboží. Nakonec v proměnné parent.zbozi[x][3] je uložena cena za jeden kus tohoto zboží.

Poté co je inicializována proměnná zboží je možné začít s vytvářením vlastního formuláře pro objednávku zboží. To je zajištěno další částí JavaScriptu která provede vlastní vytvoření tabulky a jeho zobrazení. Nejprve je nutné určit počet položek tabulky. Provede se pomocí proměnné Max do které tuto hodnotu umístil CGI skript. Poté je vytvořen cyklus pomocí příkazu for. Tento cyklus zajistí opakování bloku kódu tolikrát kolik je záznamů. Zde unádím uvedený cyklus a v něm podrobnější popis jeho funkce:

```
 var Max=$poc

 var tenp ='';

 for (count = 1;count <= Max;count++){

 tenp += '<TR>'

 tenp += '<TD>'

 tenp += '<DT><A HREF="pridano.htm"
OnClick="parent.pridej('+count+')">'

 tenp += '<IMG src="img/kosik.gif"
border="0"></A></DT>'

 tenp += '</TD>'
```

Uvedená část zajišťuje citlivost obrázku košíku na stisk myši a spuštění funkce pridej(). Funkce nejen přidá zboží do košíku, ale také zajistí informování uživatele o tom, že provedl přidání zboží. Informování se provede tak, že do pravé části se zavede dokument pridani.htm. Dokument nejprve zobrazí informaci o tom, že je něco přidáno do košíku, dále počká několik vteřin a vrátí se pomocí history.go(-1) zpět na původní stránku.

```
 tenp += '<TD>'

 tenp += '<DT>'+parent.zbozi[count][1]+'</DT>'

 tenp += '</TD>'

 tenp += '<TD>'

 tenp += '<DT><H5>'+parent.zbozi[count][2]
```

Tato část vytváří položku, ve které je umístěn podrobnější popis zboží.

```
 if (parent.zbozi[count][4] != "neni") {

  tenp += '<BR><A HREF="#"
onClick=ukaz("'+parent.zbozi[count][4]

  tenp += '")><IMG src=img/tv.gif border=0
width=30></A>'

 }
```

Tato podmínka udává jestli ke zboží není připojen doplňující obrázek, když ano, tak vytvoří v dokumentu obrázek televize. V případě stisku tohoto obrázku je vyvolána funkce ukaz() která zajistí vytvoření nového okna. V novém okně je zobrazena fotografie určeného zboží.

```
 tenp += '</H5></DT></TD>'

 tenp += '<TD>'

 tenp += '<DT>'+parent.zbozi[count][3]+'</DT>'

 tenp += '</TD>'

 tenp += '</TR>'

 };
```

Další úsek této části JavaScriptu vytvoří zobrazení ceny za jeden kus výrobku. Ještě před konečným zobrazením tohoto dokumentu je nutné vytvořit funkci zobraz. Funkce umožňuje, jak již bylo zmíněno, vytvoření nového okna a v něm zobrazení požadované fotografie určeného zboží. Deklarace funkce ukaz():

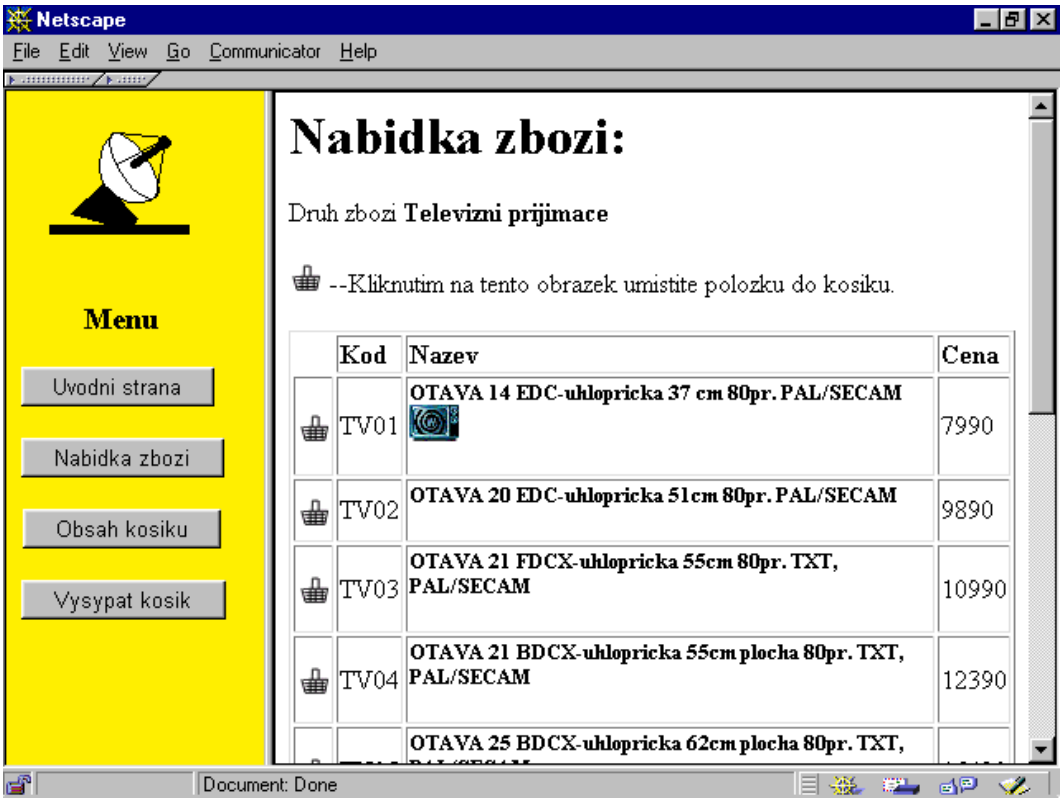
```
 function ukaz(image){
newokno=window.open('','okno','toobar=no,location=no,di
rectories=no,status
s=no,copyhistory=no,width=300,height=300')

  okno=newokno.document

  okno.close()

  okno.open()

  tmp = '<HTML><BODY><CENTER><IMG src='+image+'><BR>'

  tmp += '<FORM><INPUT type="button" value="Zavrit"
onclick=self.close()>'

  tmp += '</FORM></CENTER></BODY></HTML>'

  okno.write(tmp)

  okno.close()

 }
```

Po spuštění funkce ukaz() se nejprve provede otevření nového okna, které je celé generované JavaScriptem. Provede se pomocí příkazu window.open. Při vytvoření tohoto okna jsou mu dále zadány parametry jako velikost 300x300 bodů, žádné menu, žádné rolovací lišty, žádná stavová řádka a zákaz historie tohoto okna. Následuje už jen deklarace vlastního obsahu okna a tlačítka určeného pro jeho uzavření.

Poslední operací je vypsání proměnné tenp, ve které je uložena tabulka s celkovým zobrazením ceníku. Operace se provede pomocí metody document.write(tenp). To je celý popis souboru C.CGI který jsem rozdělil na dvě části.

1. Popis generování JavaScriptu CGI skriptem, který provádí HTML server
2. Vlastní kód JavaScriptu který zpracuje WWW klient a vytvoří tuto stránku.

Následuje zobrazení tohoto kódu tak, jak to zobrazí WWW klient.



3.4. Soubor nabidka.htm

V případě, že zákazník stiskne tlačítko Obsah košíku nacházející se v menu je spuštěn dokument se jménem nabidka.htm. Dokument zajišťuje zobrazení jednotlivých položek zboží, které zákazník vložil do košíku, dále obsahuje formulář do kterého zákazník zapíše svojí adresu, celkovou sumu za objednané zboží a na konec zajistí odeslání objednávky.

Abych objasnil celkovou funkci tohoto dokumentu uvádím zde, jeho jednotlivé části a podrobnější popis těchto částí.

```
<SCRIPT>
var tenp ='';
var kc=0;
function suma(){
 var celk=0;
 for (count = 1;count <= parent.celkem-1;count++){
 celk +=parent.nakup[count][3]*parent.nakup[count][4]
 }
 document.kusu.celkem.value=celk
 Kc=celk
 return suma
}
```

Tato funkce provede součet jednotlivých cen zboží a to tak, že vytvoří cyklus for, který prochází jednotlivé položky ceníku. Každou z těchto položek vynásobí počtem kusů a přičte k předchozí položce. Výsledek pak vloží do proměnné document.kusu.celkem.value, která představuje položku formuláře, ve kterém se zobrazí.

```
function PocKusu(akt){
 parent.nakup[akt][4]=document.kusu.elements[akt-
1].value
```

```
  suma()
 }
```

Uvedená funkce je volána při jakékoliv změně počtu kusů ve formuláři. Pracuje tímto způsobem při spuštění funkce obdrží v proměnné akt indexové číslo položky, která se změnila. Načte z formuláře hodnotu, na kterou byla položka změněna a tuto hodnotu vloží do proměnné parent.nakup a vyvolá funkci suma(), která přepočítá celkovou sumu.

```
 function Kontrola(){
 var Udaje=0
 var jmeno=document.nabidka.jmeno.value
 var psc=document.nabidka.psc.value
 var email=document.nabidka.email.value
 var count=0
 if (jmeno.length > 5) { Udaje++}
 if (psc.length > 5) {Udaje++}
 if (email.length > 5) {Udaje++}
 if (Udaje == 3) {
   document.odeslat.method="POST"
 document.odeslat.from.value=document.nabidka.email.va
lue
  document.odeslat.action="cgi-bin/mailer.cgi"
 document.odeslat.kosik.value="Jmeno:"+document.nabidk
a.jmeno.value+"\n"
 document.odeslat.kosik.value+="Ulice:"+document.nabid
ka.ulice.value+"\n"
 document.odeslat.kosik.value+="Psc:"+document.nabidka
.psc.value+"\n"
   document.odeslat.kosik.value+="E-
mail:"+document.nabidka.email.value+"\n\n"
```

```
    for (count = 1;count <= parent.celkem-1;count++){

     document.odeslat.kosik.value +=
parent.nakup[count][4]+" Kusu\n"

     document.odeslat.kosik.value +=
parent.nakup[count][1]+" "

     document.odeslat.kosik.value +=
parent.nakup[count][2]+"\n"

  document.odeslat.kosik.value +=
"Cena:"+parent.nakup[count][3]+"Kc.\n"

 document.odeslat.kosik.value+="====================
==============\n"

  }

    document.odeslat.kosik.value +=" Celkem
"+Kc+"Kc.\n"

 }else{

   alert ("Pro odeslani je nutne vyplnit vsechny
udaje")

   document.odeslat.action="nakup.htm"

   document.odeslat.method="get"

  }

 }
```

Uvedená funkce se volá v případě, kdy uživatel stisknul tlačítko pro odeslání formuláře. Proto je nutné nejdříve zkontrolovat zda jsou všechny údaje správně zadány a jestli některý z nich neschází. Do proměnných jmeno, psc a email se vloží jednotlivé hodnoty z formuláře. U každé z těchto položek se provede kontrola zda je delší než pět znaků. V případě, že ano, je zvýšena hodnota proměnné Údaje o 1. Takže v případě, že jsou jednotlivé položky dobře vyplněny musí proměnná Údaje obsahovat hodnotu 3. Jestliže to tak není je vypsané chybové hlášení „Pro odeslání je nutné vyplnit všechny údaje“a odeslání formuláře je zrušeno. Jestliže byl formulář vyplněn správně provede se nakopírování položek

zboží do formuláře, který je hned odeslán pomocí CGI skriptu mailer.cgi, který toto zajistí.

```
 for (count = 1;count <= parent.celkem-1;count++){

 tenp += '<TR>'

 tenp += '<TD>'

 tenp += '<DT><input name="poc'+count+'" size=2
value="'+parent.nakup[count][4]

 tenp += '" onChange="PocKusu('+count+')">'

 tenp += '</DT>'

 tenp += '</TD>'

 tenp += '<TD>'

 tenp += '<DT>'+parent.nakup[count][1]+'</DT>'

 tenp += '</TD>'

 tenp += '<TD>'

 tenp +=
'<DT><H5>'+parent.nakup[count][2]+'</H5></DT>'

 tenp += '</TD>'

 tenp += '<TD>'

 tenp += '<DT>'+parent.nakup[count][3]+'</DT>'

 tenp += '</TD>'

 tenp += '</TR>'};
```

Tato část pracuje stejným způsobem jako v předchozím dokumentu (C.CGI) jen s rozdílem, že na začátku je uložen počet kusů, který při změně vyvolá funkci PocKusu().

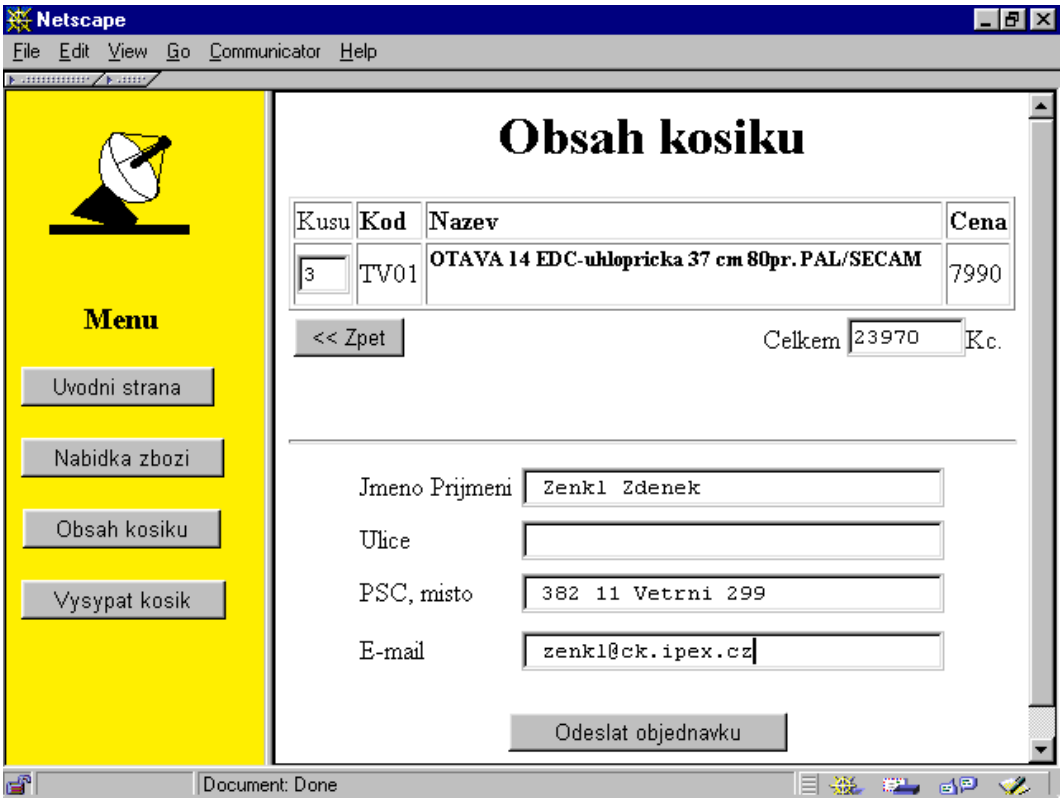
Dále jsou už jen deklarace formulářů, které zajišťují odeslání údajů o objednávce. Formuláře jsou rozděleny do dvou částí. Do prvního jsou zadány údaje od uživatele. Údaje jsou jméno, příjmení, adresa a email. Druhým je formulář, který

není na obrazovce vidět. Do tohoto formuláře jsou zapsány zpracované údaje, které jsou pak přímo odeslány. Uvádím výpis těchto dvou formulářů:

```
 <FORM name="nabidka">

 <BR>

 <CENTER>

 <HR>

 <TABLE>

  <TR>

   <TD>Jmeno Prijmeni</TD>

   <TD><INPUT name="jmeno" size=30

 onChange="parent.adresa[1]=document.nabidka.jmeno.val
ue"></TD>

  </TR><TR>

   <TD>Ulice</TD>

   <TD><INPUT name="ulice" size=30

 onChange="parent.adresa[2]=document.nabidka.ulice.val
ue"></TD>

  </TR><TR>

   <TD>PSC, misto</TD>

   <TD><INPUT name="psc" size=30

 onChange="parent.adresa[3]=document.nabidka.psc.value
"><TR>

  </TR><TR>

   <TD>E-mail</TD>

   <TD><INPUT name="email" size=30

 onCange="parent.adresa[4]=document.nabidka.email.valu
e"></TD>

  </TR>
```

```
 </TABLE>

 </FORM>

 <FORM method="POST" name="odeslat"
action="mailer.cgi">

  <INPUT name="to" type="hidden"
value="zenkl@linux.net.cz">

  <INPUT name="from" type="hidden"
value="user@linux.net.cz">

  <INPUT name="subject" type="hidden" value="Zbozi">

  <INPUT name="kosik"  type="hidden" value="text"
size=160>

  <INPUT type="submit" onClick="Kontrola()"
value="Odeslat objednavku">

 </FORM>


 suma()

 document.nabidka.jmeno.value=parent.adresa[1]

 document.nabidka.ulice.value=parent.adresa[2]

 document.nabidka.psc.value=parent.adresa[3]

 document.nabidka.email.value=parent.adresa[4]
```

Tato poslední část se spustí hned po načtení dokumentu WWW klientem a nejprve spustí funkci suma(). Nakonec vyplní jednotlivé položky, jméno, adresu a email do formuláře. Toto provede jen v případě, že tyto hodnoty byly někdy dříve už vyplněny. Takto pracuje celý tento dokument a má tuto podobu.



3.5. *Soubor mailer.cgi*

Posledním souborem, který zde budu popisovat je soubor, vlastně CGI skript, který je napsán v jazyce PERL jako předešlý. Tento CGI skript se jmenuje mailer.cgi a zajišťuje odeslání objednávky pomocí emailu. Tento skript pracuje takto:

```
#!/usr/bin/perl
print STDOUT "Content-type: text/html\n\n";
print STDOUT "<BODY>\n<title>Mail</title>\n";
```

Vytvoří nepostradatelné hlavičky, které určí, že se jedná o CGI skript.

```
for ($i = 0; $i < $ENV{'CONTENT_LENGTH'}; $i++) {
 $v .= <STDIN>;
}
```

For cyklus zajistí přečtení standardního vstupu z formuláře do proměnné $v. Na tento vstup byla odeslána veškerá data, která obsahuje druhý formulář.

```
@in = split(/&/,$v);
```

Tato řádka provede rozdělení přijatých dat na jednotlivé položky.

```
foreach $i (0 .. $#in)
 {
  @in[$i] =~ s/\+/ /g;
  @in[$i] =~ s/%(..)/pack("c",hex($1))/ge;
  @in[$i] =~ s/^.*=(.*)/$1/;
 }
```

Uvedená část provede konverzi znaků například %32 na znaky ASCII. To je nutné provést, protože WWW klient provádí při odesílání formuláře konverzi některých speciálních znaků na tvar %xx.

```
@in[3] =~ s/<BR>/ /g;
@in[3] =~ s/<br>/ /g;
```

Odstraní HTML příkaz
 a nahradí za kód konce řádky.

```
open (MAIL,"|/usr/lib/sendmail @in[0]") ||
 die "<P>Error: Nelze najit mail.\n";
print MAIL "To: @in[0]\n";
print MAIL "From: @in[1]\n";
print MAIL "subject: @in[2]\n\n";
print MAIL "@in[3]\n";
close MAIL;
```

Zde se jedná o vlastní část, která je srdcem tohoto skriptu protože provádí odeslání emailu. Dosahuje toho pomocí programu sendmail, který je součástí instalace LINUXu.

```
 print STDOUT "<BR><BR><BR><H1 align=center>Obsah
kosiku byl odeslan.</H1></BODY>\n";
```

Poslední částí je kód, který informuje zákazníka o provedení této akce.

3.6. Struktura databázových souborů

Struktura databázového souboru je řešena tak, aby byla jednoduchá a snadno čitelná nejen pro skript, ale také pro uživatele, který ji bude vytvářet, nebo opravovat. Základem této struktury jsou tři základní položky: kód zboží, popis zboží, cena zboží. Ještě je nutno podotknout, že na prvním řádku musí být jméno celé této skupiny zboží. Dále následují již jednotlivé položky. Jednotlivé části jsou odděleny znakem #. Jedinou podmínkou je aby všechny položky od dělené # byly, na jednom řádku. Kvůli složitosti zpracování dat CGI. Druhá položka může obsahovat také příkazy HTML, například
, který zajistí odřádkování textu. Dále může také obsahovat <pic.jpg>, který zajistí, že na konci textu bude zobrazen obrázek v podobě televize. V případě, že se klikne na tento symbol vytvoří se nové okno a v něm obrázek, který je uveden v těchto značkách < >. Ještě je nutno dodat že soubor musí mít koncovku TXT.

Následuje příklad zadání databáze v souboru **TV.TXT**:

```
 Televizni prijimace
TV01#OTAVA 14 EDC-uhlopricka 37 cm 80pr.<logo.gif>#7990
TV02#OTAVA 20 EDC-uhlopricka 51cm 80pr. PAL/SECAM#9890
TV03#OTAVA 21 FDCX-uhlopricka 55cm 80pr. TXT#10990
TV04#OTAVA 21 BDCX-uhlopricka 55cm plocha 80pr.#12390
TV05#OTAVA 25 BDCX-uhlopricka 62cm plocha 80pr.#16490
```

4. Stručný závěr

Jak jsem již uvedl v předu, celou práci jsem se snažil zpracovat tak, aby ji bylo možno použít v běžné praxi. Proto jsou možná některé její části příliš podrobné. Budu spokojen bude-li tato moje diplomní, práce (třeba přes tvrdé kritické připomínky) přijata.

5. Seznam použité literatury.

[1] Mark C. Reynolds: JavaScript - Profesionální řešení: UNIS publishing

[2] Matt Welsh a Lar Kaufman: Pou□íváme LINUX: Computer Press

[3] Chip special: HTML @ Java : Vogel

[4] Zdroje nalezené na INTERNETu

[5] Osobní poznámky z přednášek.

6. Příloha

Disketa s programem prezentace firmy s formulářem na objednávku zboží.