ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA

V PRAZE

Provozně ekonomická fakulta

Obor Informatika

World Wide Web aplikace

Autor Bakalářské práce:

Miloš Veselý

Vedoucí bakalářské práce:

Ing. Pavel Šimek, Ph.D

Čestné prohlášení

Prohlašuji, že jsem svou bakalářskou práci na téma World Wide Web aplikace vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené bakalářské práce prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 28. listopadu 2011 …….

Miloš Veselý

Poděkování

Rád bych touto cestou poděkoval vedoucímu bakalářské práce Ing. Pavlu Šimkovi, Ph.D. za cenné rady, připomínky, metodické vedení práce a pomoc při korektuře. Poděkování patří také všem, kteří mě při tvorbě bakalářské práce podporovali, zejména mé rodině.

World Wide Web aplikace

World Wide Web applications

Anotace

Bakalářská práce se zabývá porovnáním dvou nástrojů používaných k programování interaktivních webových aplikací v JavaScript a Java applet. Zaměření je zejména na porovnání dvou vytvořených aplikací vzhledem k nejpoužívanějším webovým prohlížečům. Uvádí nejvíce používané nástroje pro tvorbu webových stránek. Popisuje vytváření základních skriptů a programů. Poukazuje na problémy s bezpečností www aplikací.

Annotation

Bachelor´s thesis deals with comparation of two tools used to programme an interactive web applications in JavaScript and Java applet. The focus is particularly on comparing two created applications within five most used web browsers. Lists the most used tools used for web pages development. Describes creating of basic scripts a programmes. Points at problems with safety of www applications.

Klíčová slova

internet , www aplikace, http, HTML, javascript, php, css, xml, java, java applet, activeX, microsoft .NET,

Keywords

internet , www aplikace, http, HTML, javascript, php, css, xml, java, java applet, activeX, microsoft .NET,

Obsah

1. Úvod

Vznik webových aplikací je spojen se zrozením internetu. První zmínky o internetu jsou z konce 60. let v Americe. Nebyl to zcela internet, tak jak je znám dnes, ale spíše větší množství počítačů napojených do sítě. Název této sítě byl ARPANET obsahoval 4 uzly a velice pomalou rychlostí spojoval výpočetní střediska. Jeho použití bylo pouze pro armádní účely. Převrat nastal s příchodem vysokorychlostních modemů, které umožnily komunikaci pomocí klasické telefonní linky. V roce 1984 používaly internet pouze americké univerzity a armáda. V obou institucích bylo dohromady přibližně tisíc počítačů. Jedna z hlavních změn nastala v roce 1986, kdy vzniká síť NFSNET, páteřní síť internetu v Americe. Tato síť nahradila ARPANET , který se vrátil k výhradně armádním účelům. Z internetu se stala otevřená doména pro vzdělávání a výzkum. Postupně se do internetu připojily všechny americké univerzity a výzkumné ústavy. Zpočátku byl internet pouze pro nekomerční použití, jakožto akademická síť. Americká vláda byla přesvědčená, že internet by mohl pomoci americké ekonomice, proto začal proces privatizace a komercionalizace internetu po celé USA. Nejvíce se angažoval americky viceprezident Al Gore za podpory většiny politiků. Byla vybudována nová gigabitová síť NREN (National Research and Education Network). Proces privatizace končí v roce 1993 a od té doby je internet rozdělen na akademický a komerční. V tomto roce počet uživatelů překonal hranici jednoho miliónu. Skutečným startem byl tedy rok 1993, kdy vznikl internetový programovací jazyk HTML současně s prvním prohlížečem a pojmem World Wide Web.

Jednou z hlavních postav, které se zasloužili o rozvoj internetu, je Dr. Douglas C. Engelbert. Byl první, kdo přišel na myšlenku provázaných dokumentů, neboli hyperlinků a dokonce i k její realizaci na mainframe počítačích. Současně to je také vynálezce počítačové myši. Bohužel doba byla nezralá a proto jeho vynálezy nebyly oceněny. V 80. letech přišel Ted Nelson s projektem Xanadu, který byl na konci 80. let financován firmou Autodesk. Xanadu byl projekt, jenž jasně definoval rozprostřené síťové prostředí, které bylo velmi podobné dnešnímu modernímu webu. Nelson byl první, kdo použil slovo hypertext. Stejně jako Engelbert, tak i Nelson přišli se svými projekty příliš brzo a proto byli neúspěšní. Ve švýcarském výzkumném středisku CERN v Ženevě, které bylo jedno z mála připojeno k rodícímu se internetu, definoval Tim Berners-Lee hypertextový systém.

Byl to první intranet na světě. V roce 1990 Berners-Lee napsal první program pro tvorbu primitivních hypertextových stránek a navrhl název „World-Wide Web“. Poté následují roky nepřetržitého růstu, který místy dosahuje rozměrů epidemie. Již zmíněný Berners-Lee se v roce 1994 stává ředitelem WWW Consorcia(W3C), ve kterém jsou lidé z CERN. Jejich činnost spočívá ve vytváření standardů.

Z počátku byly webové stránky statické a jednosměrné, což znamená, že neposkytovaly zpětnou komunikaci s uživatelem. Proto vznikají www aplikace, které se zaměřují na oživení stránek a na možnost odezvy od serveru směrem k uživateli. Moderní web obsahuje běžící programy, které využívají skripty vytvořené v jazycích JavaScript, Java applet, dynamické HTML (DHTML), ActiveX. [5, 12]

2. Cíl práce a metodika

Hlavní cíl práce se zaměřuje na porovnání jazyků JavaScript a Java, přičemž u Javy se jedná o applety, které se využívají pro webové aplikace. Dílčím cílem bakalářské práce je analýza webových aplikací. Jejich rozdělení dle druhu a způsobu použití. Vysvětlení základních pojmů spojených s vytvářením webových stránek a aplikací. Uvedení jejich výhod a nevýhod současně s možností jejich použití. Nebyla opomenuta ani na krátká historie, jak internetu tak aplikací. Dále je zmíněná bezpečnost aplikací, která je v dnešní době často probíraným tématem a možnosti řešení těchto problémů. Práce okrajově uvádí popis a vytváření základních skriptů, jenž jsou později využity pro praktickou část práce.

Pro vytvoření práce byla použita následovná metodika. Na základě analýzy z odborné literatury jsou zde popsány nejrozšířenější nástroje pro tvorbu webové aplikace, které jsou v dnešní době nedílnou součástí internetu. Základem praktické části je vytvoření jednoduchých skriptů ve dvou různých jazycích a následné vyhodnocení výsledků na základě komparace aplikací. Dané skripty jsou vytvořeny z poznatků, které byly získány během tvorby této práce. Dále budou tyto skripty testovány na nejrozšířenějších webových prohlížečích. Jelikož se v dnešních dobách mění zastoupení nejoblíbenějších prohlížečů, je jím věnována kapitola, která se zaměřuje na jejich postavení na trhu a změny v posledních letech. Jsou zde uvedeny i rozdíly v užívání prohlížečů v ČR a ve světě.

Závěr vlastní práce spočívá v porovnání náběhových časů daných skriptů na jednotlivých prohlížečích. Z tohoto porovnání vyplynou informace spojené s klady a zápory skriptů. Testování bude probíhat na vytvořené webové stránce, která využívá pro test skript, který vypočítává načtení stránky s danou aplikací. Pro objektivnost je tato metoda testování zavedena na obě vytvořené aplikace. Výsledky jsou zaznamenány a posléze porovnány s ohledem k webovým prohlížečům. Pro názornost jsou výsledky porovnání zobrazeny pomocí grafů.

3. Skriptovací a programovací jazyky pro WWW aplikace

3.1 Rozdělení

Programování aplikací lze rozdělit do dvou koncepcí. První z nich je stavové programování, které umožňuje systému přecházet mezi různými stavy. Každý z těchto stavů nabízí odpovídající funkce. Oproti tomu událostmi řízené programování má systém stále ve stejném stavu. Systém je stále ve stavu čekání, jakmile se dočká podnětu, použije nejlepší způsob, jak ho obsloužit a opět se vrací do stavu čekání. Ve většině případů se využívá druhý systém, jelikož je flexibilnější.

Dále je možné programovací jazyky rozdělit na *interpretované* a *překládané*. Jazyky interpretované jsou rovnou čteny, aniž by potřebovaly kompilátor. Jejich výhodou je, že jsou čitelné téměř pro všechny systémy. Druhý typ, jazyky překládané, využívají speciální programy, které je převádějí do formy čitelné pro daný systém a hardware. V praxi nejsou rozdíly skoro znatelné, protože řada moderních interpretů jazyk průběžně překládá. Často se používají i kombinované přístupy, kdy je potřeba překladače, jenž převádí program do speciální formy, která není vhodná přímo pro hardware, ale je připravená pro jiný program, který umožní spuštění. Typickým zástupcem tohoto přístupu je jazyk Java.

Odlišné rozdělení je na jazyky *deklarativní* a *imperativní*. Postupy, při nichž se vysvětluje počítači, co přesně má udělat, se označují imperativní. Naproti tomu při deklarativním programování se počítači přesně popíše, jak má výsledek vypadat. To, jak se k němu dojde, je v zásadě na daném počítači, ne v rukou programátora. Bohužel při deklarativním způsobu není řešení, jak se systém dobere k výsledku, vždy zrovna praktické či přímo vhodné. Proto je tento jazyk velmi málo užíván a jeho jediným úspěšným zástupcem je Prolog. [2]

3.2 HTML

Předchůdcem jazyka HTML pro vytváření internetových stránek byl SGLM jazyk Charlese Goldfarba. Dnes je HTML nejznámější jazyk, který spadá pod záštitu W3C konsorcia. Základ HTML vychází z SGML. Tyto jazyky byly založeny na principu co se

má udělat, a ne jak se to má udělat(*imperativní přístup)*. Tenhle způsob měl největší výhodu v přenositelnosti na jakýkoli operační systém, platformu nebo uživatelské rozhraní. Vývoj HTML probíhal od roku 1989, ale zásadní byl rok 1996, kdy byl schválen standard HTML 3.2, jenž měl přímého nástupce HTML 4.0, který byl o rok později schválen. Tato verze byla plnohodnotná, jelikož vývojáři spojili své síly a pracovali na vývoji společně. Tím docílili doposud nejcelistvější verze, která byla až dodnes používána jako standard. Dodnes platí verze 4.0 , ke které bylo postupem času vytvořeno spoustu rozšíření. Verze HTML 5.0 už okupuje svět, ale její podpora není zdaleka zakomponována do všech prohlížečů. Jazyk HTML má danou syntaxi, ale nezměrné množství variabilních úprav. HTML dokument je možné vložit na internet i přesto, že zcela nesplňuje pravidla správného webu, což znamená, že obsahuje chyby. Jelikož je jazyk HTML textového formátu, není nikdy kompilován do binární či jakékoli podoby. Znamená to, že vytvořená stránka je okamžitě interpretována jakýmkoli prohlížečem bez potřeby dalších úprav. Odlišnost je v binárních souborech, jako jsou obrázky či zvuky, které jsou vkládány pomocí odkazů. Problémem, jenž nastává při tvorbě v HTML je, že různé platformy zpravidla nemusí podporovat různé jazyky, v tomto případě se dává k dispozici online překladač kódování. V dnešní době je tento problém vyřešen prohlížeči, jenž mají podporu většiny jazyků zakomponovanou v sobě.

Základním stavebním kamenem HTML je TAG neboli značka. V podstatě se jedná o chráněné slovo jazyka HTML, které se uzavírá do špičatých závorek. To, co se vyskytuje mezi závorkami je příkaz jazyka HTML. Vše co je mimo ně je vlastní obsah stránky. Tagy jsou rozděleny do dvou skupin a to podle toho, zda je tag párový či nepárový. Párové tagy jsou ty, které používají jak počáteční tak konečný tag. Rozdíl je v lomítku v konečném tagu. Text, který se nalézá mezi je vlastní obsah s přiřazenou hodnotou od tagu. V případě nepárových tagů je zásadní rozdíl, že neobsahují zakončující tag s lomítkem. Vše se napíše do špičatých závorek současně s parametry, které se vztahují k pouze jednomu elementu. Jsou to například elementy jako pozadí, obrázek či vodorovná linka. Zprvu je důležité rozmyslet si, jak vlastně bude stránka vypadat a postupně pomocí dané struktury tvořit požadovaný cíl. [5]

3.3 JavaScript

Nejpoužívanější jazyk pro dynamické stránky, vzhledem k jeho jednoduchému použití, pomocí vložení skriptu do HTML kódu. Jednou z nejdůležitějších vlastností JavaScriptu je, že se jedná o programovací jazyk, jehož činnost probíhá v prohlížeči návštěvníka stránky a ne na serveru, kde jsou stránky uloženy. Jinak řečeno je JavaScript interpretovaný na straně klienta. Pro programátory je tento způsob interpretace dosti omezující, protože z důvodu bezpečnosti není možné zapisovat a načítat údaje ze souborů.

I přes významné omezení okruhu možností je tento jazyk hojně využíván pro interaktivní webové stránky. V podstatě je jeho užití pouze doplňkové a proto ho nikdy nevyužijeme k vytváření celistvých www stránek, ale pouze k vylepšení už hotových stránek. Jeho využití je hlavně v aplikacích jako jsou hodiny, dotazníky a podobné aplikace vyskytující se na webových stránkách. Při vytváření nebo začleňování skriptu do HTML stránky jsou dva způsoby jak to provést. Prvním je vložit skript přímo do HTML kódu s použití odvolání na něj. Druhá možnost je napsat odkaz na soubor obsahující daný skript. Následná funkčnost skriptu je totožná v obou případech až na rozdíl v rychlosti načítání, která je lehce nižší v samostatném souboru z důvodu menší velikosti HTML kódu. [4, 9]

3.4 PHP

Jedná se o skriptovací jazyk, který je zabudován na straně serveru a je optimalizován pro prostředí www. Jelikož jde o skript, provádí se uvnitř dokumentu HTML. Podpora PHP je ve většině operačních systémů, od systému Unix po operační systémy Microsoft Windows. PHP umožňuje práci s různými WWW servery, přičemž webový server Apache je hlavním spojencem. Spolu vytvářejí prostředí, které díky ceně a nízkým nárokům na hardware, umožňuje vývoj rozsáhlých webových aplikací. Protože je možné provozovat oba produkty na různých platformách operačních systémů, je snadné vytvořenou aplikaci implementovat do většiny prostředí. Nedílnou součástí je i podpora velkého množství databází a protokolů pro elektronickou poštu. Jelikož se jedná o objektově orientovaný jazyk má PHP plnou podporu práce s třídami a objekty.

Provedení PHP skriptu zabezpečuje server. Nejdříve přijme požadavek na dokument se skriptem od prohlížeče, poté ho předá na zpracování interpretu PHP.

Z interpretu vychází vygenerovaný dokument, který obsahuje pouze kód HTML. Pokud je soubor správně zpracován, interpret ho předá zpět serveru, který jej odešle prohlížeči. [5]

3.5 CSS

Technologie nazvaná CSS neboli Cascading Style Sheets se používá k vytvoření stylu webové stránky. Pomocí CSS je možné určovat barvu, písmo, velikost písma, rámeček, podtržení, tučnost atd. a tím ovlivňovat vzhled celého webu z jednoho souboru. V HTML je potřeba definovat styl pro každý odstavec, což znepříjemňuje práci, jak programátorům, tak uživatelům, jelikož se zvětšuje objem souboru HTML. Výhoda CSS spočívá v naformátování několika stylů, na které se poté odkazuje v celém textu. Poprvé se styly objevili v Internet Exploreru 3.0 a tak jako spousta dalších novinek, které se později staly standardem, byly uvedeny společností Microsoft. Později se styly objevily v HTML 4.0 a byly zde rozšířeny a doplněny. Prohlížeče, které byly uvedeny na trh v roce 1997 v drtivé míře obsahují podporu CSS a od doby , kdy je platná specifikace CCS 2.1, jsou styly podporovány ve všech prohlížečích. Stejně jako JavaScript, tak i CSS je možné mít uložené mimo stránku tj. externě. [5]

3.6 XML

Historie XML úzce souvisí s SGML, což je univerzální značkovací metajazyk. Vše začalo potřebou standardu pro výměnu údajů mezi různými počítači ve velkých průmyslových firmách. V roce 1986 vznikl standard ISO 8879, který obsahuje definici jazyka SGML. Byl to první standard, který povolil oddělit data od jejich zpracování. „*Na základě analýzy struktury dat se vytváří slovník označovaný jako DTD (Document Type Definition)*“. Jde o druh slovníku, který naznačuje obsah jednotlivých objektů s přesně definovanou syntaxí. Tyto slovníky se mohou lišit, protože různé datové objekty mohou mít různé množiny údajů. Dokument SGML se tvoří jako textový dokument, který obsahuje prvky oddělené značkami, které byly nastaveny ve slovníku DTD. Jakožto text je dokument přenositelný na různé systémy, které mají implementovaný analyzátor dokumentů SGML. Analyzátor načte dokument a pomocí slovníku DTD a vlastních značek v dokumentu umožní určit strukturu a zpracovat obsah. I přes jeho pokrokový způsob tento jazyk neuspěl z důvodu náročnosti na analyzátory, které byly ve své době drahé a pomalé. Jeho úspěch spočívá v jazycích, které jsou dnes užívány a jejich základ pochází právě z SGML. Samozřejmě hlavními zástupci jsou HTML a XML. [4, 8]

3.7 Java a Java applety

V roce 1995 společnost Sun Microsystems vydává jazyk Java. Ve stejném roce vzniká i JavaScript. Původní název Javy byl Oak, který byl popsán jako zjednodušená verze C++. Hlavními přednostmi Javy jsou:

- Dynamické propojování funkcí programu do běžící aplikace pomocí převzetí ze sítě.
- Nezávislost na platformě CPU.
- Záruka bezpečnosti vůči virům.

Java se zpočátku velmi rychle šířila mezi programátory v podobě Java appletů. Applety poskytovaly plnohodnotné programovací schopnosti na straně klienta, aniž by byla ohrožena bezpečnost. Pro spuštění Java aplikace je potřeba mít JVM modul, který se musí zvlášť nainstalovat. Java applety též potřebují JVM, které je v dnešní době součástí téměř všech prohlížečů. Applety mohou být externí soubor na který se odkazuje z HTML nebo se vkládají přímo do HTML kódu, což znamená, že samostatně jsou nefunkční na rozdíl od Java aplikací. Z hlediska ukládání do mezipaměti, technikám toku dat a chytrému programování, jsou applety velmi podobné jako standardní aplikace systému.

Javu lze použít i pro psaní servletů, jenž se používají pro generování obsahu HTML stránek v rámci webového serveru. Trend se ale ubíral jiným směrem a to od tlustého ke štíhlému klientovi. Applety zastupují tlustého, protože zahlcují klienty různými doplňky, které je třeba nainstalovat. JavaScript se vyvíjí cestou štíhlého klienta, čímž si zaručuje oblíbenost a tloustnutí nechává na serveru. [1, 10, 11]

3.8 ActiveX

Alternativa k Java Appletům je tvorba prvků v jazyce ActiveX, který se z uživatelského hlediska příliš neliší od Appletů. Rozdíl je znát hlavně v technologickém pozadí. Prvky ActiveX jsou většinou malé programy umístěné na serveru, odkud se stahují do prohlížeče, kde jsou spouštěny. V rámci dokumentu HTML tvoří ActiveX samostatný objekt reprezentovaný specializovaným tagem a to <OBJECT>. Je to objektový jazyk, proto lze definovat vlastnosti a metody objektu, ke kterým se přistupuje pomocí externího programového kódu, např. JavaScriptu. Nevýhodou ActiveX je závislost na platformě

Windows a jeho nutná instalace do klientského počítače, což přináší riziko ohrožení bezpečnosti. Tento problém stojí za menším množstvím užívání této technologie na webových stránkách. Přitom jeho plus je, že stačí stáhnout pouze jednou, oproti appletům, které se stahují pokaždé při návštěvě stránky. Poté, co je stažen, se nainstaluje a není třeba ho nikdy více stahovat. Jediný případ, kdy je nutné opětovné stažení, je v případě nové verze. K programování aplikací v ActiveX se používá Visual Basic, jenž pochází také od Miscrosoftu. [11]

3.9 Microsoft .NET

Jazyk pocházející z dílny Microsoft. Na svět přišel v roce 2001, kdy vyšla beta-verze. Vynikající prostředek pro vývoj webové aplikace, webové služby a dalších aplikací fungujících na internetové platformě.

„*Jedním ze základních rysů prostředí .NET Framework je jeho jazyková nezávislost*.“

Jak již bylo řečeno v .NET je možné programovat pomocí jazyka C#, C++, JScript, Visual Basic nebo COBOL. Síla .NET spočívá v neomezených možnostech, které nabízí. Vše se vyvíjí ve společném jazykovém běhovém modulu neboli CLR(Common Language Runtime) a knihovně tříd rámce .NET (Framework Class Library – FLC). CLR slouží pro abstrakci služby operačního systému a je to v podstatě vykonávací jádro pro řízené aplikace, jejichž všechny akce podléhají schválení u CLR. Aplikace psané v rámci .NET naschvál vypouští API Windows a další nástroje a technologie místo nich se pracuje s FCL. Pokud se zavolají funkce API Windows nebo objekt COM je potřeba přejít z řízeného kódu, který běží pod CLR na neřízený, jenž běží bez pomocí runtimového modulu v podobě nativního strojového kódu. Tyto přechody omezují výkonnost a mohou být dokonce zakázány správcem systému.

V 90.letech měla největší potenciál služba ASP.NET. Byla to revoluce ve webovém programování zvaná Active Server Pages – ASP, která nabídla snadno použitelný model dynamického vytváření obsahu HTML na webových serverech pomocí skriptů. Nástupce ASP je zmiňovaná ASP.NET, jenž nabízí zcela nový způsob psaní webových aplikací, který se ničemu nepodobá. [8]

4. Popis vytváření základních skriptů

4.1 JavaScript

Pro většinu programovacích jazyků je základním stavebním kamenem proměnná, taktéž i pro JavaScript. Pod názvem proměnná se skrývají dočasné paměťové buňky, které mají za úkol uskladnění určitých druhů informací pro pozdější zpracování. Jejich rozdílnost spočívá v druzích údajů, jenž jsou do proměnných ukládány. Jsou to číselné údaje, znakové údaje, logické údaje a programátorem vytvořené vlastní proměnné, u kterých se zvolí název a příslušná hodnota. Samotná realizace přiřazení hodnoty k proměnné se provádí pomocí jednoduchého rovnítka takto:

```
Nazev_promenne=hodnota

Anebo

Nazev_promenne=cislo
```

Příklad č.1: Přiřazení proměnné

Poté se s proměnnou jedná jako s číslem, tím pádem se může sčítat, odčítat, násobit, dělit. Jinak řečeno umožňuje používat většinu operandů. V případě seskupení většího počtu proměnných, které spolu úzce souvisí, vznikne pole.V podstatě se dá říci, že je to druh proměnné. Výhoda použití pole spočívá v jednoduché manipulaci s více proměnnými a v možnosti relativního odkazování na prvky, které tvoří pole. Při vytváření pole je nejprve nutné ho deklarovat, neboli dát programovacímu jazyku najevo vznik tohoto pole. Tato deklarace se provádí takto:

```
Nazev_pole=new Array()

Anebo

Nazev_pole=new Array(prvek1, prvek2….prvekN)
```

Příklad č.2: Deklarace pole

V poli může být neomezené množství prvků, přičemž každý tento prvek má své pořadové číslo. Pomocí tohoto čísla se může na daný prvek v poli odkazovat.

Při tvorbě skriptů je potřeba znát funkce, které usnadňují operace, jenž se provádí během vytváření. Namísto zcela zbytečného a zdlouhavého psaní toho samého se napíše funkce, která to provede ve chvíli, kdy se zavolá. Přesná definice zní: „*Jde o určité seskupení příkazů v programu, které je nečinné, dokud je někdo neaktivuje – nevyvolá.*“ Pro funkce je nejlepší, když se vyskytují v hlavičce, díky tomu se funkce při načítání stránky stihne nahrát do paměti prohlížeče, což vyloučí chybové hlášení z důvodu volání nezcela nahrané funkce. Funkce lze rozdělit na funkce bez argumentu, u kterých není možné ovlivnit průběh úlohy a funkce s argumentem, kde tato možnost existuje. Při tvorbě funkce bez argumentu se nejprve napíše její název, poté do složených závorek souhrn příkazů, které se provádí, jakmile je funkce zavolána. To, co se vyskytuje mezi složenými závorkami se nazývá tělo funkce. Obecný tvar funkce bez argumentu je:

```
Function nazev_funkce ()

{

tělo  funkce }
```

Příklad č.3: Funkce bez argumentu

Více variabilnější je funkce s argumentem, jež umožňuje určitý druh regulace. Pomocí argumentů se upravuje, již vytvořená funkce, dle požadavků programátora. Funkce může obsahovat více různých argumentů, tím se stává pružnější, ale zároveň více ovlivňuje výsledek její činnosti, což nemusí být vždy přínosné. Zápis funkce s argumentem je:

```
Function nazev_funkce (arg_1, arg_2….arg_n)

{Tělo funkce}
```

Příklad č.4: Funkce s argumentem

Arg_1 až arg_n jsou argumenty funkce.

Tyto dva druhy funkcí po provedení zobrazí daný výsledek pomocí textu. Existují také funkce, které nemají zcela takto viditelný výsledek. Mohou to být funkce zaměřené na vnitřní účely např.: získávání informací o uživateli či zpracovávání číselných údajů.

JavaScript stejně jako většina programovacích jazyků obsahuje několik typů rozhodovacích konstrukcí. Pomocí těchto konstrukcí se vytváří vnitřní logika, která posuzuje výkon skriptu, který dále určí, co se má provést. K posuzování se používají porovnávací operátory. Jejich úkolem je vracet logické hodnoty (pravda/nepravda) podle výsledku porovnání. Při jejich užití je možnost porovnávat nejen čísla, ale i další druhy proměnných. Nejjednodušší rozhodovací konstrukcí s pevně určenou úlohou je přiřazovací podmínka. Ta vyhodnocuje, zda je řešený problém pravdivý či nepravdivý, a na základě toho přiřazuje danou hodnotu do proměnné. Zapisuje se ve tvaru:

```
Proměnná=(výraz) ? h1:h2
```

Příklad č.5: Přiřazovací podmínka

Když je výraz pravdivý, do proměnné se vybere hodnota h1. Pokud je výraz nepravdivý, proměnná bude mít hodnotu h2. Lehce složitější variantou je podmínka IF …ELSE, která umožňuje podle pravdivosti výrazů provádět určitý odlišný sled příkazů. Je to nejpoužívanější a nejpružnější typ rozhodovací konstrukce. Její zápis je:

```
If (výraz) Proveď A

Else Proveď B
```

Příklad č.6: Podmínka If - Else

Jestliže se v závorce nachází pravdivý výraz, provede se část skriptu A. Pokud je výraz nepravdivý, provede se část skriptu B. Příkaz ELSE není povinný, jeho použití je dle potřeby. Kombinací předchozích konstrukcí je podmínka SWITCH. Od ostatních se liší pouze tím, že je činnost vyhodnocuje hodnoty konkrétně zadané proměnné, namísto vyhodnocování pravdivosti logických výrazů.

Obecný zápis je:

```
Switch (proměnná)
{
Case h1 : proveď A;
Case h2 : proveď B;
Default : proveď C;
}
```

Příklad č.7: Podmínka Switch

Pokud se nevybere ani jedna z hodnot h1 a h2 provede se příkaz default, jenž je označen jako část C. Většinou se současně s touto podmínkou používá příkaz BREAK, který zastaví provádění skript. Kdyby se ve skriptu tento příkaz nenacházel, výpis z této funkce by se skládal ze všech zadaných podmínek včetně defaultní.

Pro zjednodušení určitých příkazů se hojně využívají logické operátory. Pomocí operátorů AND, OR nebo NOT si programátor ulehčuje řešení funkcí.

Výrazným přínosem JavaScriptu na webových stránkách je jeho interaktivita, která nabízí návštěvníkům možnost ovlivnit dění na internetových stránkách. Tato vlastnost otevřela dveře programátorům ve vytvoření propojení mezi stránkou a návštěvníkem. Nejzákladnější formou této komunikace jsou dialogová okna. Některá je možno vyvolat a použít k vlastním účelům, jiná zas dokáže vyvolat pouze samotný prohlížeč stránky. Z toho vyplývá základní členění:

- Systémová dialogová okna

Do této skupiny patří všechny typy dialogových oken, ve kterých se nachází chybová hlášení a bezpečnostní varování. Bohužel JavaScript neobsahuje nástroje pro práci s nimi.

- Uživatelská dialogová okna

Mají za úkol informovat návštěvníka, získávat od něj konkrétní informace a potvrzovat činnosti. Jsou to výstražná a informační okna, potvrzovací okna Ok/Storno a vstupní okna. Jejich výhoda spočívá v jednoduché ovladatelnosti a práce s nimi v JavaScriptu. [9]

Pokud chce programátor upozornit uživatele na důležitou informaci, která se ztrácí ve větším množství textu na obrazovce, je nejvhodnější využít výstražné informační okno. V podstatě na návštěvníka vyskočí ihned po načtení stránky, čímž okamžitě upozorní na danou informaci. Obecný tvar je:

```
Window.alert("text");
```

Příklad č.8: Výstražné informační okno

K zalomení řádku v okně se používá výraz \n.

Dalším způsobem, jak komunikovat s návštěvníky je možnost jednoduché otázky s odpovědí ano či ne. Tato možnost se využívá při pokládání otázek, na které je jasná odpověď. Toto okno se nazývá potvrzovací a jeho skript je následovný:

```
Windows.confirm("text")
```

Příklad č.9: Potvrzovací okno

U tohoto typu okna je lehce složitější konstrukce, jelikož je zde použita volba dvou logických hodnot. Záleží jakou si návštěvník vybere, pokud volí *ok*, vrátí skript logickou hodnotu *true*. V případě opačné volby *storno*, vrátí se logická hodnota *false*.

Posledním druhem těchto oken je vstupní okno, které umožňuje návštěvníkovi zadávat konkrétní hodnoty na zadané otázky. Tyto odpovědi mohou být např. číselný údaj nebo slovní spojení. Využití těchto odpovědí se nachází dále ve skriptu, např. pokud je na stránky povolen vstup pouze lidem starším 18 let, je ve skriptu napsána podmínka, která povolí či zamítne vstup podle hodnoty zadané návštěvníkem.

Obecný tvar konstrukce k zobrazení vstupního okna je:

```
Windows.prompt("text","implicitní text");
```

Příklad č.10: Vstupní okno

Místo, kde je napsáno *text* slouží pro vložení otázky pro návštěvníka. To, co se napíše místo *implicitní text* se zobrazí ve vstupním poli tohoto okna při jeho objevení. Samotný vzhled okna je pevně dán, proto jediná možnost jak ho upravit, je naprogramovat si své vlastní okno.

Příkazy cyklu je další z mnoha funkcí JavaScriptu ulehčující život programátorům. Jejich hlavním úkolem je opakovaně provádět určitou část skriptu, dokud platí zadaná podmínka. Dají se využít i v případě, kdy se zcela neví, kolik opakování se má realizovat. Rozlišujeme tři druhy příkazů cyklu:

1. *Do ... while*
2. *While*
3. *For*

Příkaz *do ... while* provádí opakování určité části skriptu, dokud platí zadaná podmínka. Specifické pro tento příkaz je, že se provede minimálně jednou, a to i v případě, že zadaná podmínka neplatí. Obecný tvar příkazu je:

```
Do
{
Tělo cyklu
}
While (výraz)
```

Příklad č.11: Příkaz Do - While

Rozdíl mezi předchozím a následujícím výrazem *while* je, že k provádění cyklu dojde jenom tehdy, pokud je zadaný výraz pravdivý. Toto je jediná diference, jinak jsou tyto výrazy ekvivalentní. Základní tvar výrazu je:

While (výraz)

```
{
Tělo cyklu
}
```

Příklad č.12: Příkaz While

Posledním cyklickým výrazem je *For*. Tento příkaz se odlišuje od předchozích cyklů tím, že realizuje určitý konečný počet opakování. Musí být dopředu jasně znám počet opakování. Obecný tvar příkazu je:

```
For (inicializace proměnné; podmínkový výraz; aritmetická operace)
{
Tělo cyklu
}
```

Příklad č.13: Příkaz For

Pod pojmem *inicializace proměnné* se skrývá přiřazení počáteční hodnoty do řídící proměnné. *Podmínkový výraz* je podmínka, která udává, dokdy má cyklus probíhat. V neposlední řadě *aritmetická operace* určuje, která operace se má provést na řídící proměnné při každém opakování cyklu. [9]

4.2 Java

Pro vytváření programů v jazyce Java je potřeba sadu Java SE Development Kit (JDK) a prostředí pro tvorbu, což může být např. NetBeans IDE. Jedná se vývojářské prostředí, které vám odstraní problémy s konfigurací, jež jsou běžné pro kompilátor *Javac*.

Základní koncepce se skládá z objektů, tříd, dědičností, rozhraní a balíčků. Primárním klíčem jsou objekty, jak už napovídá technologie objektově orientovaného programovaní. Objekt je možné rozdělit na *stav* a *chování.* Pro porozumění je jednodušší využít objekt z reálného světa např. pes má určitý stav (jméno, barvu, rasu) a chování (štěkání, aportování, vrtění ocasem). Stav je v objektu ukládán do *datových složek* (field- v některých jazycích proměnná) a poskytuje své chování pomocí *metod* (v některých jazycích funkce). Metody slouží jako primární mechanismus pro komunikaci mezi objekty a využívají k tomu práci s vnitřním stavem objektu. V případě, že objekt skrývá svůj vnitřní stav, požaduje, aby všechny interakce probíhaly pomocí metod, označuje se to jako zapouzdření dat, což je základní princip OOP.

Více objektů stejného druhu se označuje *instance třídy objektů*. Třída sama o sobě představuje plán, podle kterého vznikají jednotlivé objekty. Užitečná vlastnost, díky ní je možné zmenšit obsah kódu, je *dědičnost*. Třídy umožňují dědit své vlastnosti ostatním podtřídám. Vždy může existovat jedna nadtřída, která má neomezený počet *potomků,* neboli podtříd. Dalším nespecifikovaným typem je rozhraní, jenž komunikuje s vnějším světem. V obvyklé podobě to je skupina souvisejících metod bez implementace. Jinak řečeno rozhraní představuje smlouvu mezi třídou a okolím, přičemž dodržování této smlouvy kontroluje kompilátor. Posledním z uvedených je balíček. Zastupuje jmenný prostor, který organizuje sadu souvisejících tříd a rozhraní. V podstatě to je prvek, který uspořádává jednotlivé třídy a rozhraní, jenž spolu souvisí do souboru. Programy napsané v jazyce Java mohou obsahovat tisíce tříd, proto je rozumné vytvářet organizační balíčky.

Nedílnou součástí jazyka Java jsou proměnné. Jak již bylo řečeno proměnné je možné jinak nazvat jako datové složky, ale neplatí to pro všechny. Lze je také následovně rozdělit:

- **Instanční proměnné** (nestatické datové složky)

Objekty jsou ukládány do nestatických datových složek neboli do složek, které jsou deklarované bez klíčového slova *static*. Název instanční vypovídá o vlastnosti, že všechny hodnoty jsou jedinečné pro každý objekt.

- **Proměnné třídy** (statické datové složky)

Tyto proměnné mohou být libovolná datová složka deklarovaná s modifikátorem *static*. Pro kompilátor značí slovo static právě jednu kopii této proměnné bez ohledu na to, kolik instancí třídy bylo vytvořeno.

- **Lokální proměnné**

Slouží pro ukládání stavu metod do proměnných. Při deklarování je důležité dodržet umístění, jelikož syntaxe je stejná jako u datové složky, záleží pouze na místě, kam se zapíše. Její místo je mezi otevírací a uzavírací závorkou metody, proto jsou tyto proměnné viditelné pouze pro metody, ve kterých jsou deklarovány.

- **Parametry**

Většinou se volí za parametr řetězec *args* nebo *argv*, i když je možné zvolit jakýkoli řetězec. Parametr se určuje pro metody a vždy se klasifikuje jako proměnná nikoli jako datové složky.

Základní datové typy

Jazyk Java obsahuje silnou typovou kontrolu. Proměnné je proto zapotřebí před použitím deklarovat. Součástí deklarace je uvedení typu a názvu proměnné např.:

```
Int prevod = 1;
```

Příklad č.14: Deklarace proměnné

Takto lze sdělit programu, že existuje datová složka s názvem *„převod“*, která uchovává číselná data a jejíž výchozí hodnota je „1“. Datový typ určuje hodnoty, jenž se mohou do proměnné ukládat a operace, které je možné s proměnnou provádět. Jazyk Java podporuje 8 základních datových typů:

- **Byte** (8bitové celé číslo se znaménkem uložené jako doplněk dvou. Hodnoty se pohybují od -128 do 127. Využívá se hlavně kvůli úspoře paměti ve velkých polích.)

- **Short** (16bitové celé číslo se znaménkem uložené jako doplněk dvou. Jeho hodnoty se pohybují od -32 768 do 32 767. Vyplatí se ze stejného důvodu jako byte, ale má větší rozsah.)

- **Int** (32bitové celé číslo se znaménkem uložené jako doplněk dvou. Nejčastěji používaný typ s hodnotami od -2 147 483 648 do 2 147 483 647. Je dostatečně veliký a neexistuje důvod zvolit jiný, až na úsporu paměti (viz výše).)

- **Long** (64bitové celé číslo se znaménkem uložené jako doplněk dvou. Má ještě větší rozsah než int a to – 9 223 372 036 854 775 808 až po 9 223 372 036 854 775 807.)

- **Float** (32bitové číslo s plovoucí desetinnou čárkou s jednoduchou přesností. Neměl by se používat pro přesné hodnoty jako finanční částky. Jeho vymezení je definováno normou IEEE 754.)

- **Double** (64bitové číslo s plovoucí desetinnou čárkou s dvojitou přesností. Jedná se o výchozí typ jako int, ale podobný float s rozdílem ale většího rozsahu, který je opět definován normou IEEE 754.)

- **Boolean** (Tento datový typ má pouze dvě hodnoty *true* a *false*. Je reprezentován jedním bitem informace s velikostí, která není přesně definována. Používá se pro podmínky typu pravda či nepravda.)

- **Char** (Umožňuje uložit jeden 16bitový znak v kódování Unicode. Jeho nejnižší hodnota je 0 a nejvyšší 65 535.)

Proměnná *String* sice nepatří mezi základní typy, přesto je hojně využívána. Jedná se o proměnnou definující znakové řetězce.

Pokud je nutné vytvořit proměnnou, do které je možné uložit více hodnot , jedná se o *pole*. Je to objekt typu kontejner, který uchovává pevný počet hodnot stejného typu. Jeho délka se určuje při jeho vytvoření a zůstává pevná.

Syntaxe pro vytvoření pole je následovná:

```
int[] pole;

pole = new int[2]

pole[0] = 100;

pole[1] = 200;
```

Příklad č.15: Deklarace pole

Výše je definováno pole typu int, které obsahuje 2 prvky. Do hranaté závorky se udává počet hodnot, jenž pole obsahuje. Když je potřeba dostat se k určitému prvku z pole využívá se indexování, které začíná od nuly.

Příkazy, jež upravují kód, jsou obvykle vykonávány v pořadí odshora dolů, ve kterém jsou uvedeny. Příkazy řízení toku ovšem mění pořadí pomocí rozhodování, cyklů a větvení. Program potom určité bloky kódu realizuje podmíněně. Základní příkaz řízení toku, jenž je nejjednodušší je *if – then.* Testuje příslušnou část kódu a následně provádí pouze tehdy je-li hodnota *true.* Syntaxe kódu je obdobná jako v jiných jazycích a zní:

```
If (podmínka) {

Příkaz;}
```

Příklad č.16: Příkaz if–then

Pokud je podmínka vyhodnocena jako *true* provede se příkaz. V opačném případě přejde řízení na konec příkazu. Lehce složitější je příkaz *if-then-else*, který poskytuje druhou větev provádění. Pokud je podmínka vyhodnocena jako *false* provede se příkaz *else.*

Syntaxe tohoto příkazu je podobná předešlé a to:

```
If (podmínka) {

příkaz;}

else{

příkaz;}
```

Příklad č.17: Příkaz If-then-else

V tomto případě je jasné pokud se neprovede první příkaz provádí se druhý. Jestliže je potřeba velké množství větví k provádění je nejlepší využít příkaz *switch*. Tělo příkazu *switch* se označuje jako přepínač, protože je možné ho označit jedním nebo více návěštími *case* či *default*. Příkaz *switch* využívá příkazu *break*, aby mohl zastavit řízení po podmínce a neprocházel skrze další alternativy. Syntaxe příkazu *switch* je :

```
Int název = podmínka;

Switch (název) {

Case 1: Co se má provést.; break; }
```

Příklad č.18: Příkaz switch

Tento příkaz má výhody zejména pro *break,* která může být využita a nemusí. Pokud je potřeba, aby řízení procházelo skrze část bloku a poté se zastavilo a udělalo výpis je vhodné použít výraz *break*.

V případě, že je potřeba trvale provádět blok příkazů, dokud má daná podmínka hodnotu *true*, využije se příkazu *while* nebo *do-while*. Syntaxe příkazu *while* je následovná:

```
While (výraz) {

Příkaz nebo příkazy }
```

Příklad č.19: Příkaz while

Jedná se o vyhodnocování hodnoty typu *boolean*. Dokud se vrací v bloku *while* hodnota *true* provádí se příkaz stále dokola, až do chvíle, kdy se vrátí hodnota *false.* Druhý příkaz *do-while* je odlišný v tom, že vyhodnocuje testovací výraz na konci cyklu, nikoli na jeho začátku. Z toho vyplývá, že příkazy v bloku *do* se vždy provedou alespoň jednou. Jeho syntaxe podobná:

```
Do {

Příkaz nebo příkazy

} while (výraz);
```

Příklad č.20: Příkaz do-while

Příkaz *for* je vhodný pro procházení hodnot v určitém rozsahu. Často je označován jako *cyklus for* kvůli tomu, že opakovaně prochází cyklus do splnění určité podmínky. Obecná forma příkazu *for* zní:

```
For (inicializace; ukončení; inkrementace) {

Příkaz nebo příkazy }
```

Příklad č.21: Příkaz for

Vysvětlení tohoto příkazu je následovné:

- Výraz *inicializace* značí cyklus, který se provede jednou na začátku provádění cyklu.
- Pokud je výraz *ukončení* vyhodnocen jako *false*, cyklus se ukončí.
- Poslední výraz *inkrementace* je vyvolán po každém průchodu cyklu a umožňuje zvyšování či snižování hodnoty.

Předností tohoto výrazu je možnost deklarovat proměnnou přímo v rámci cyklu. Pokud daná proměnná není potřeba mimo cyklus je nejlepší ji deklarovat v inicializačním výrazu. Tímto opatřením je možné omezovat životnost a předcházet chybám. Pokud je potřeba vynechat aktuální průchod cyklem *for, while* nebo *do-while* využívá se k tomu příkaz *continue*. Ten přejde na konec těla nejvnitřnějšího cyklu a vyhodnotí výraz typu *boolean,*

který cyklus řídí. Poslední z příkazů větvení je příkaz *return*, který zajistí opuštění aktuální metody a řízení toku se vrátí tam, odkud byla metoda vyvolána. Příkaz *return* má dvě formy, přičemž jedna z nich vrací hodnotu a druhá nikoli. Rozdíl při psaní syntaxe příkazu *return* spočívá v přidání výrazu nebo hodnoty přímo za slovo *return*. [14]

5. Přednosti a negativa skriptů a programů

5.1 Program

Pojem počítačový program v České Republice není žádným způsobem výslovně definován, což znamená, že neexistuje žádný autorský zákon ani směrnice o právní ochraně počítačových programů. Toto vymezení nebylo zakomponováno úmyslně, z důvodu brzkého zastarání takovéto legislativní úpravy kvůli technickému pokroku. Přesto v Americe tato legislativní definice počítačového programu existuje a zní takto: „*A computer program is a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result.*“ Tuto definici lze přeložit jako: „*Počítačový program je soubor příkazů a instrukcí pro použití přímo nebo nepřímo v počítači za účelem dosažení určitého výsledku.*“ Česká definice počítačového programu dle právní teorie je následující: *„Soustava příkazů, které jsou schopny řídit činnost počítače za účelem dosažení konkrétního výsledku.“* Zásadní rozdíl, jenž rozlišuje program od skriptu je nutnost kompilace. Každý program se musí zkompilovat do binárního kódu, aby byl čitelný pro počítač. [13]

5.2 Skript

Program, který není zapotřebí kompilovat se nazývá skript, i přesto se může jednat o zdrojový kód. Hlavní výhody skriptu jsou v možnosti vnoření do HTML souboru. V České Republice neexistuje právní forma definice skriptu, proto lze uvést pouze vysvětlení na pochopení o co se vlastně jedná. Skript je posloupnost příkazů, které mohou být spuštěny programem nebo skriptovacím strojem bez zásahu uživatele. Jeho využití je zejména na webových stránkách, ale i v rámci operačního systému. [5, 17]

5.3 Bezpečnost webových aplikací

Bezpečnost webových aplikací může být ohrožena interakcí webové aplikace a prohlížeče zejména HTTP, HTML a JavaScript. Jak již bylo vysvětleno dělení http na metody, tak i při útoku záleží jaká metoda je právě používána.

První typ útoku je SQL Injection, jenž využívá špatného ošetření vstupu od uživatele při sestavování dotazů a parametrů do jiných systémů. Tento problém lze snadno vyřešit nesestavováním SQL dotazů pomocí jednoduchého skládání řetězců. Potíže spojené

s SQL se netýkají pouze http, ale i dalších dotazovacích systémů. Jelikož je http bezestavový protokol, i přesto je potřeba alespoň část stavu určitou formou přenášet k tomu se využívá mechanizmus session (sezení). Při prvním přístupu se uživateli vytvoří sezení a jeho identifikace se zapíše pomocí cookie nebo pomocí předávání v URL. V tomto spočívá i zranitelnost http, protože mechanizmus sezení je možné zneužít. Je k tomu potřeba pouze získat identifikátor sezení, který umožní korektní přihlášení uživatele. Tudíž je nutné generovat identifikátory sezení zcela náhodně. Pokud využijete uložení identifikátoru sezení do URL je nezbytné zajistit, aby URL nebylo dostupné žádnému jinému serveru jako http Referer. Tento Referer slouží k informování, ze které stránky uživatel přišel.

S rozvojem webových aplikací, jež využívají JavaScript přišel i nový typ útoku JavaScript Hijacking. Jedná se o aplikace využívající JavaScript pro výměnu dat. V podstatě jde o zneužití formy *json*, ve které je možné najít vlastnost email. V případě, že *json* podporuje callback, což znamená, předání všech dat funkci, která je zavolána, stačí pouze nadefinovat tuto funkci a poté sdílí data mezi aplikacemi na různých doménách. Jiných druh, který také potřebuju JavaScript, aby mohl útočit je clickjacking. Jedná se o schované tlačítko, o kterém uživatel nemá ponětí a omylem ho zmáčkne. Tím způsobí instalaci viru či v lepším případě se pouze odkáže na stránku, která se schovává pod tímto tlačítkem. Často je toto tlačítko ukryté v iframe. Zamezení je možné spíše přes prohlížeč než přes samotnou webovou aplikaci nebo zajištěním aplikace proti vložení do iframe.

Mezi nejhorší útoky patří Cross Site Scripting a Cross Site Request Forgery. První ze jmenovaných zneužívá nahrazování HTML klíčových znaků za jejich entitní vyjádření. Řešením tohoto problému je použití takových systémů, které rovnou převádí výpis textů na obrazovku do podoby entitních vyjádření. Text, který nemá být převeden je potřeba explicitně označit. Pokud by se označoval text, jenž má být převeden, je riziko, že se přehlédne nebezpečné místo a tím se způsobí chyba XSS. V případě druhého útoku hrozí riziko v odlišné URL, o které nemá uživatel tušení a na kterou vstoupí bez jakéhokoli vědomí. Tento útok může být vložen do HTLM např. jako URL cíl obrázku, přičemž ve skutečnosti se jedná o formulář nebo potvrzení nějaké aplikace. [3]

5.4 Http protokol

Jedná se o aplikační protokol, který funguje na systému požadavek/odpověď. Klient pošle požadavek, neboli zadá cestu ke stránce a server mu zpětně odpovídá. Základním stavebním kamenem http je URL, jenž identifikuje dokument poskytnutý uživateli. Http je bezestavový protokol, což znamená, že neumí uchovávat stav komunikace, dotazy spolu nemají souvislost. Ve verzi 1.0 http ještě nebylo spojení trvalé, ale pomocí příkazu *keep-alive* server neukončil spojení, nýbrž čekal na další požadavek. Od verze 1.1 jsou všechna spojení trvalá.

Protokol také nabízí řešení pro přístupová práva a ověřování totožnosti. Tato funkce je v http implementována, není třeba vymýšlet vlastní řešení zabezpečení. Princip je jednoduchý, klient pošle požadavek na nějaké URL, server vrátí stavový kód 401. Poté si prohlížeč vyžádá po klientovy přihlašovací údaje, které znovu pošle na server. Zbývá pouze ověřit přihlašovací data a poté server odpovídá požadovanou stránkou. Autentizace pomocí http má své výhody mezi ně patří např. použití hotového řešení, standardní dialog a v neposlední řadě řešení API, které umožňuje přístup na stránku jinému systému než uživateli s prohlížečem. Zároveň existují i nevýhody, jenž je nemožnost přizpůsobovat přihlašovací dialog. V případě opisování obrázku CAPTCHA nebo blokování práce v jiných panelech prohlížeče. Pro bezpečnost komunikace je potřeba využívat šifrování. Nejčastější způsob šifrování je zabalený http protokol do SSL/TLS běžící standardně na portu 443. Mezi další vlastnosti http patří možnost komprese dat, která se nejvíce ocení u textových formátů typu (X)HTML, CSS, JavaScript. Komprese zmenšuje objemy přenášených dat a tím šetří částečně procesorový čas serveru. [7]

6. Porovnání technologií JavaScript a Java

Pro názornou komparaci byly vytvořeny dvě totožné aplikace v jazycích JavaScript a Java. Jedná se o dialogová okna, která vyskakují při navštívení stránky. První okno se dotazuje na rok narození, ve druhém se vám zobrazí odpověď, zda máte dostatečný věk na návštěvu stránky. Pokud jste starší 18 let můžete pokračovat na dané stránky. V opačném případě je vám přístup odepřen. Většinou se tato opatření objeví na stránkách, jejichž obsah je nepřístupný pro mladistvé. Vytvořené skripty by měly demonstrovat rozdíly mezi použitými jazyky a zároveň ukazovat výhody a nevýhody spjaté s danými aplikacemi. Obě aplikace byly vytvořeny jako externí soubor. Stránky na kterých jsou testovány, obsahují stejné kódy. V případě Java jazyka, je zde popsán vytvořený applet s funkci *If* a *else*.

Obě aplikace byly vytvořeny v programu PSPad editor, který se specializuje na tvorbu www stránek a aplikací. Tento program ulehčuje vývoj tím, že označuje párové tagy, podporuje úpravy více souborů najednou, obsahuje konverzi mezi znakovými sadami a vytváří hlavičky k daným jazykům. Je to volně dostupný program, který pochází od českého tvůrce Jana Fialy. Na jeho internetových stránkách www.pspad.cz, je možné stahovat rozšíření v podobě uživatelských skriptů.

6.1 Aplikace v JavaScriptu

První popis je zaměřen na strukturu JavaScriptu, jenž je v současnosti více využíván. K tvorbě aplikace byly použity zmíněné skripty v kapitole 4.

```
1. <script LANGUAGE="JavaScript" TYPE="text/javascript">
2. rok=window.prompt("Zadejte rok narození:","");
3. aktualni_rok=2011;
4. vek=(aktualni_rok - rok);
5. if  (vek>=18)
6. { povoleni="Mužete vstoupit.";
7. window.alert("Je Vám "+vek+" let.\n"+povoleni); }
8. else
9. { povoleni="Nejste plnoletý a proto odejděte z této stránky.";
10. window.alert("Je Vám "+vek+" let.\n"+povoleni);  }
11. </script>
```

Příklad č.22: Aplikace v JavaScriptu

Aplikace pracuje na jednoduchém principu s využitím podmínky. Syntaxe jazyka JavaScript je přehledná a lehce pochopitelná. Na příkladě je vidět použití dialogových oken, které se spouští při otevření stránky. Pomocí rovnítka je přiřazena hodnota nebo funkce k proměnné, se kterou se později provádí podmínka. Pokud je podmínka splněna, provádí se první funkce. V opačném případě se přechází na *Else*, jenž provede funkci druhou.

6.2 Aplikace v Java Applet

```
import java.awt.Graphics;
```

```
import javax.swing.*;
public class demo extends JApplet {
            int   Vek;
            int  rocnik;
  public void init() {
   String stari;
          stari=JOptionPane.showInputDialog ("Zadejte rok narození.");
          Vek=Integer.valueOf(stari) ;
   rocnik=(2011 - Vek); }
public void paint (Graphics g){
  int a = 18;
  int b = rocnik;
 if(b >= a)


           JOptionPane.showMessageDialog(null, "Je Vám " + b + " let.\n Mùžete vstoupit.",
"Rozhodnutí", JOptionPane.PLAIN_MESSAGE);
          else {
          JOptionPane.showMessageDialog(null, "Je Vám " + b + " let.\n Nemùžete
vstoupit.", "Rozhodnutí", JOptionPane.PLAIN_MESSAGE); }}}
```

Příklad č.23: Java Applet

Jak lze vidět na kódu appletu je zde složitější syntaxe, proto je zde podrobněji popsána. Před umístěním na webové stránky je nutno kód zkompilovat pomocí nástroje JVM. Při vytváření kódu je zprvu potřeba vytvořit si třídu a definovat metodu, poté se deklarují proměnné. Jelikož se jedná o applet je zde využita metoda *JApplet*. Jako datový typ byl použit Integer, který představuje celá čísla se znaménkem. Pouze proměnná *stari* je String, protože obsahuje řetězec.

První okno, které vás přivítá na stránce je Input, což znamená, že do něj vkládáte vstupní data v podobě roku narození. Jednoduchá rovnice vám vrací počet let a zapisuje tento údaj do proměnné ročník. V kódu se poté vyskytuje stejná podmínka jako

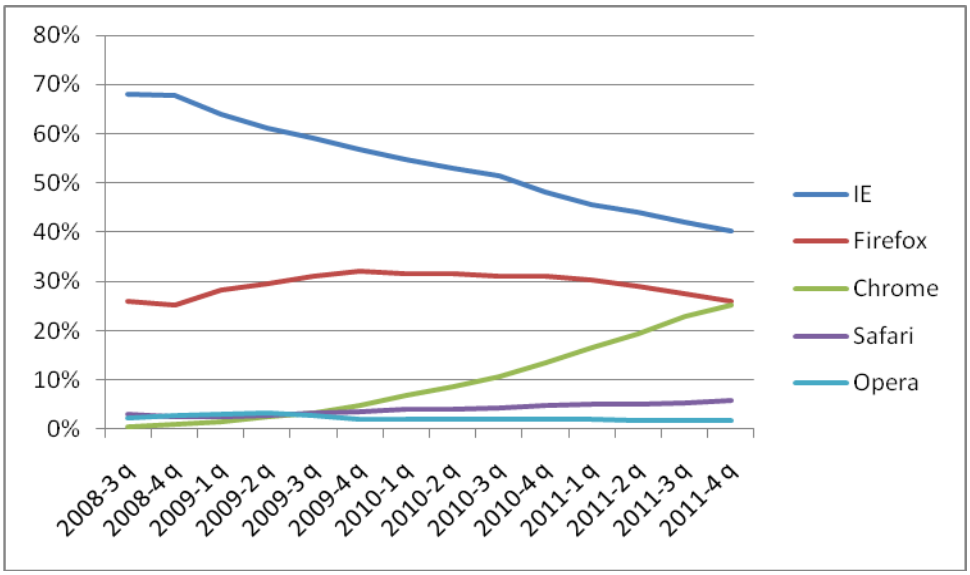
v předešlém JavaScriptu. Jedná se o podmínku *If – Else,* jenž zajistí okno, které se vrátí na základě vašeho věku. Pro zobrazení výsledných oken je třeba deklarovat novou metodu *paint*. Pro vyhodnocení podmínky je použit základní operand větší nebo rovno. Finální rozhodnutí zaleží na věku návštěvníka. Pokud je mladší 18 let je mu přístup odepřen. V případě staršího návštěvníka, je možno pokračovat v prohlížení stránek.

6.3 Porovnání aplikací

Z předchozího popisu je zřejmé, že JavaScript je jednodušší jazyk pro vytváření podobných aplikací. Není třeba přiřazovat datový typ k proměnné stejně tak se nemusíte starat ani o metody. Z pohledu zápisu je kód jasně čitelnější i pro oko laika. Mezi další nevýhody Java Appletů je nutnost kompilace pomocí JVM. Ohledně podmínky *If-Else* je zde stejná syntaxe, podoba v obou kódech a výsledek aplikace je tím pádem totožný. Výhoda Java spočívá v chytrém kompilátoru, který ukáže na chyby v kódu. Vypíše přesně řádek s chybou a nenechá program zkompilovat, dokud se chyby neopraví. Přičemž u JavaScriptu jste odkázáni sami na sebe v hledání chyby. Jak bylo řečeno výhody a nevýhody Appletů jsou v JVM, který je chytrý, ale zároveň jen komplikuje život neustálým kompilováním vytvořených aplikací.

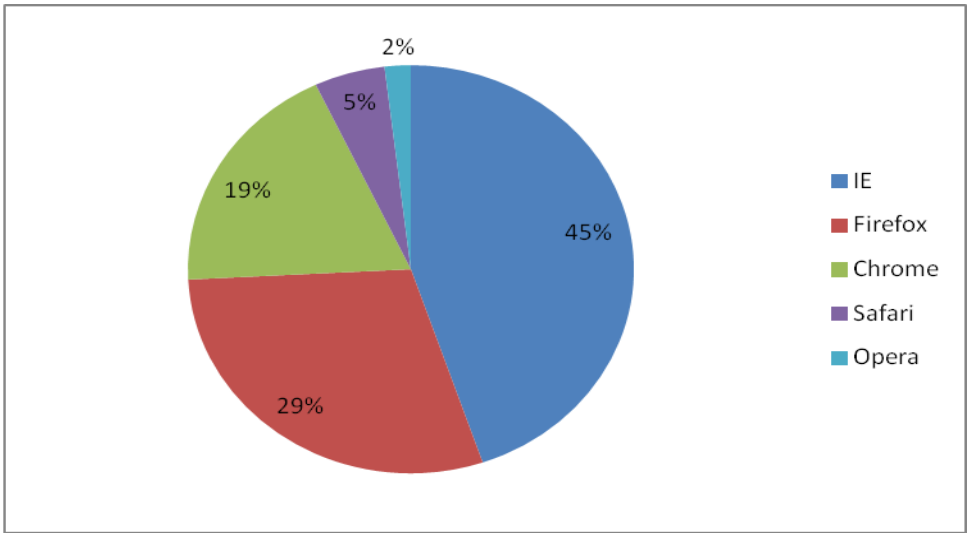
6.4 Webové prohlížeče

Podobně jako vývoj internetu, tak i vývoj prohlížečů šel velmi rychle. Zásadní věcí, jež ovlivnila vývoj prohlížečů, je HTML. Prohlížeč interpretuje jazyk HTML a je proto důležité dodržovat standardy, jako u kódů, aby se klient neztrácel. Z počátku se výrobci snažili zakomponovat do prohlížečů nestandardní podporu věcí, které byly teprve v návrhu nebo které si sami vymysleli. V poslední době se tento trend určitým způsobem uklidnil. Většina prohlížečů se snaží dodržovat určité standardy a tím pádem interpretovat HTML stejně jako ostatní. Nejrozšířenější prohlížeče v dnešní době jsou Internet Explorer, Mozilla (Firefox), Webkit (Chrome), Opera a Safari. Samozřejmě největší zastoupení má Explorer, ale v posledních letech se jeho pozice otřásá. S příchodem Chromu se celkově trh s prohlížeči zamíchal. Jeho rostoucí tendence je neuvěřitelná, což je názorně vidět na následujícím grafu. [6]

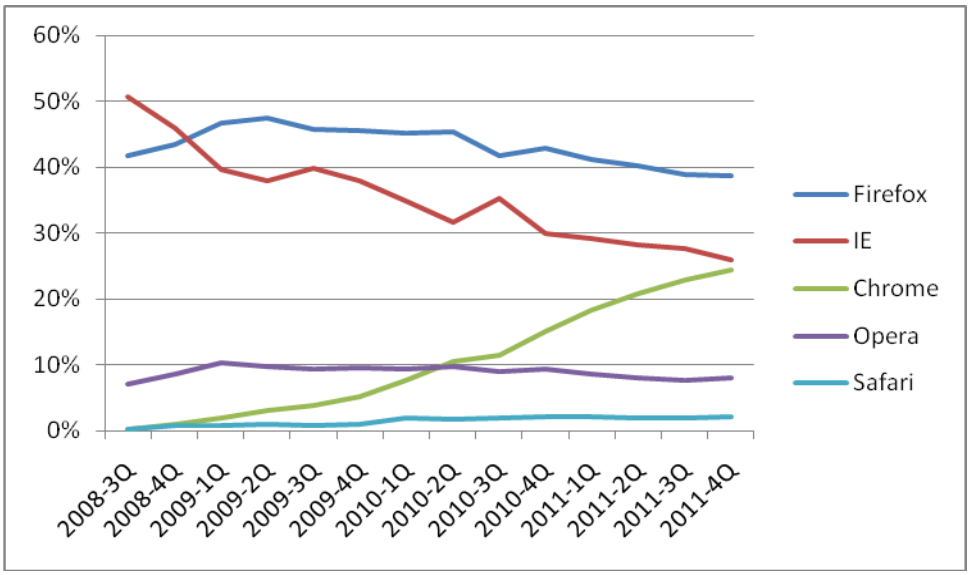


Graf č.1: Zastoupení prohlížečůve světě [16]

Z grafu lze vypozorovat prudký pokles Internet Exploreru, který prožívá trpké období, jelikož na konkurenci nebyl zvyklý. Explorer měl od začátku velkou výhodu, protože je dodáván s operačním systémem Windows, je to první prohlížeč, se kterým se setkají klienti Microsoftu. Tento fakt markantně ovlivňoval postavení Exploreru, ale v současnosti jsou uživatelé náročnější a vybíravější, proto obliba Exploreru rychle klesá. Oba tyto grafy jsou zaměřeny na průzkum ve světovém měřítku. Velké rozdíly lze nalézt v České republice, kde je skladba zastoupení zcela odlišná od světa.



Graf č.2: Procentuální zastoupení ve světě v roce 2011 [16]



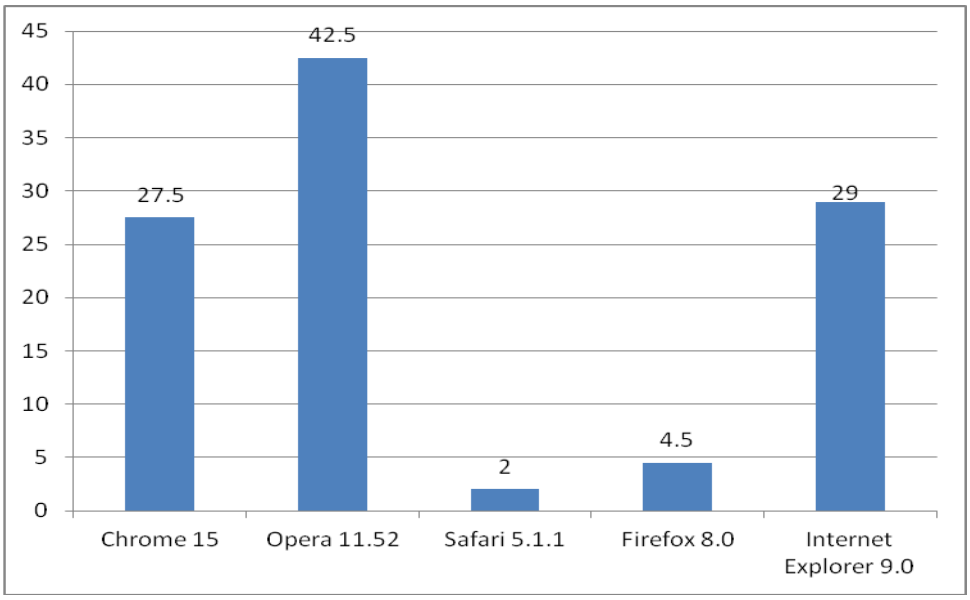
Graf č.3: Zastoupení prohlížečů v ČR [16]

Jak je vidět na grafu výše, zastoupení prohlížečů v ČR je zcela odlišné. Od roku 2009 je nejvíce používaným prohlížečem v ČR Firefox. Zatimco Explorer má stejně jako ve světě, tak i v ČR klesající trend. Chrome svým tempem, pokud bude dále pokračovat, se dostane před Explorer a zaujme druhou pozici na českém trhu. Další zajímovosti je postavení Opery k Safari, jež se drží nejníže a jehož růst je velmi sporadický. Z hlediska světového je to obráceně, jelikož prodej počítačů Apple vzrůstá, tím pádem roste i zastoupení prohlížeče Safari. I když ma Firefox počet klientů kolem 39% jeho postavení pomalu slábne. Předběžné prognózy jsou nejvíce na straně Chrome, který raketově uhání vzhůru za prventsvím mezi prohlížeči.

6.5 Porovnání prohlížečů

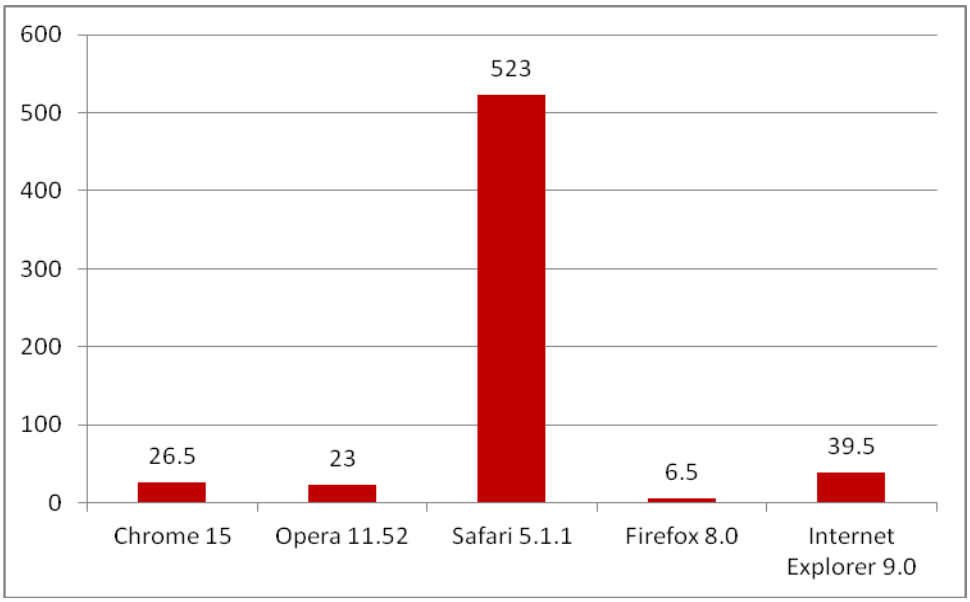
Všechny z těchto prohlížečů podporují jak JavaScript tak Java Applety a testy, jež budou prováděny, se zaměřují na rychlost načtení těchto aplikací. Některé z prohlížečů obsahují nástroje pro vývojáře, které mají zakomponovanou funkci *TimeLine*, ze které lze jednoduše vyčíst potřebné údaje. Jelikož tato funkce není dostupná pro všechny druhy prohlížečů, které budou testovány, je vhodnější použít webovou aplikaci Page Loading Script [15], jenž vypisuje časy načtení dané stránky s aplikací a je dostupná na www.dreamincode.net. Tímto způsobem testování je zajištěna objektivnost. Tato data budou poté statisticky porovnána a cílem bude určit, jaký prohlížeč je vhodný pro určité aplikace.

Testy probíhají na operačním systému Windows 7, jenž se nachází na osobním počítači značky Asus, na který bylo nainstalováno všech pět zmiňovaných prohlížečů. Připojení k internetu je zajištěno pomocí UPC 25 Mb/s download a 1,5 Mb/s upload. Za účelem testování byla vytvořena www stránka script-test.wz.cz, na které jsou umístěny obě aplikace a je možné si je vyzkoušet. Jelikož se časy pokaždé měnily, což mohlo způsobit čekání na server, jsou aplikace desetkrát po sobě otestovány a ze zjištěních časů je získán medián, se kterým se dále počítá. Z důvodu častých odchylek byla dána přednost mediánu před průměrem, který by byl zkreslený. Výslédné časy načtení JavaScriptu jsou zobrazeny prostřednictvím grafu níže s hodnotami v milisekundách.



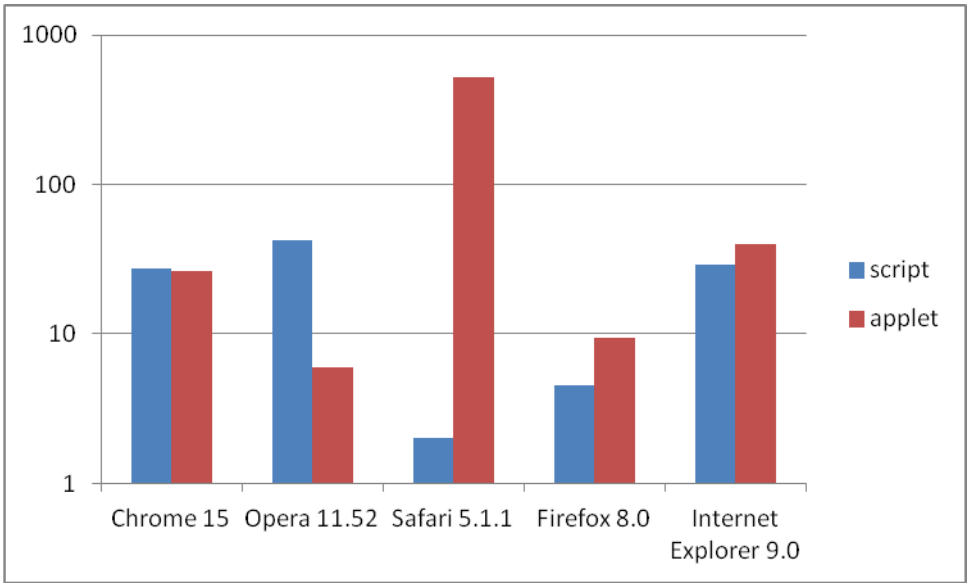
Graf č.4: Načítací doby JavaScriptu v milisekundách

Z grafu lze zřetelně vyčíst, že prohlížeč Safari zvládl načíst JavaScript v nejkratší době. Lehce pomalejší byl Firefox, který se stále pohyboval v časech do jedné desítky milisekund. Poté už s výrazným odstupem byl prohlížeč Chrome následován Internet Explorerem a Operou. Časy jsou velmi odlišné, což je způsobeno principem, kterým prohlížeče načítají stránky. Tento princip se odvíjí od použitého jádra, což představuje hlavní podstatu prohlížeče. Ve zde testovaných prohlížečích lze nalézt jádra Webkit, Gecko, Presto a Trident.



Graf č.5: Načítací doby Java Appletu v milisekundách

Zatímco během testu JavaScriptu byl prohlížeč Safari chválen za minimální čas, nyní se dá říci, že vyhořel. Jelikož se Applety stahují do prohlížeče při prvním načtení, zde se stahoval pokaždé, což způsobilo prodlevu a nakonec i nejhorší čas načtení. U ostatních prohlížečů nebylo, ani první načtení tak zdlouhavé. U Safari je už na první pohled znatelný rozdíl v načtení JavaScriptu a Java appletu.

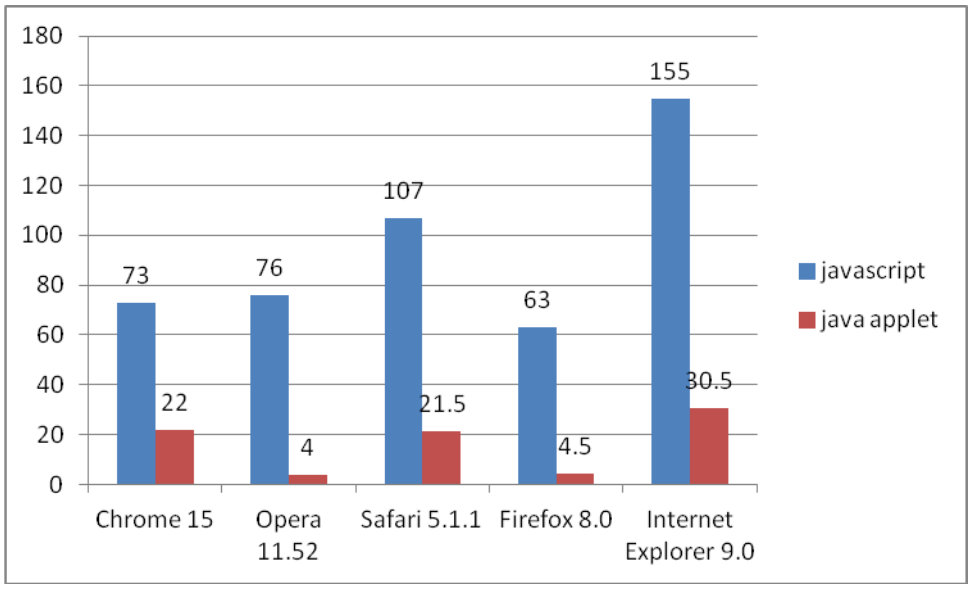


Graf č.6: Porovnání načítacích časů aplikací v milisekundách

Z pěti testovaných prohlížečů, JavaScript nejrychleji načítaly prohlížeče Safari, Firefox Internet Explorer. Naopak Java applety byly lépe načteny prohlížeči Chrome a Opera.

Nejmenší rozdíl mezi načtením JavaScriptu a Java appletu zaznamenal prohlížeč Chrome, tento rozdíl byl pouze 3,8 %.

Předešlé aplikace byly jednoduché a nemusely zcela pravdivě vypovídat o vlastnostech prohlížečů, proto z důvodu kvalitnějšího porovnání, bylo provedeno druhé měření. Opět jsou porovnány aplikace napsané v JavaScriptu a v Java appletu s tím rozdílem, že tentokrát se jedná o kalkulačky se základními funkcemi. Jejich kód je k nahlédnutí v příloze. Dané kódy byly získány z internetových stránek. Kalkulačka napsaná v JavaScriptu je z části vytvořená v HTML, jež definuje výsledné zobrazení jako formu tabulky, ve které jsou tlačítka, která se odkazují na funkce napsané v JavaScriptu. Java applet je napsán v jazyce Java, ve kterém lze určit všechny, jak grafické tak matematické funkce.



Graf č.7: Porovnání načítacích časů kalkulaček v milisekundách

Výsledné časy jsou opakem ve všech prohlížečích včetně Safari oproti prvnímu testování. Ve všech případech byl rychlejší Java applet. Načítací doba JavaScriptu byla u všech testovaných prohlížečů o 58% – 96% delší než u Java appletu. Za znatelnými rozdíly se pravděpodobně skrývá, již zmiňovaný HTML kód použitý k zobrazení kalkulačky v JavaScriptu. Samostatný skript napsaný v JavaScriptu je mnohem kratší, čitelnější a logičtější než kalkulačka vytvořená v Java appletu.

7. Závěr

Bakalářská práce popisuje historii a nástroje pro tvorbu interaktivního webu. Zabývá se jazyky, aplikacemi a skripty, které je možné využít pro vytváření webových aplikací. Práce se soustředí na porovnání dvou základních nástrojů sloužících k tvorbě interaktivních webových stránek na straně klienta, a to JavaScript a Java applet.

Z porovnání těchto dvou jazyků lze říci, že každý se hodí pro různé aplikace, přičemž ani jeden nemá takové schopnosti, aby mohl plně zastoupit druhého. Prostřednictvím Java appletu je možno programovat složitější aplikace, vytvářet kvalitnější grafiku nebo modelovat fyzikální zákony. Existence kompilátoru se může na první pohled jevit jako těžkopádná, na druhou stranu programátorovi pomáhá odladit vytvořený kód.

JavaScript je čistě specializován pro webové aplikace. S tím jsou spojeny určité výhody a omezení pro tvorbu takzvaně živého webu. Kód je psán v podobě skriptu, který zaručuje jednoduchý způsob vkládání na webové stránky a usnadňuje provádění oprav. Syntaxe JavaScriptu je poměrně jednoduchá a uživatelsky přívětivá. Za výhodu JavaScriptu lze považovat i fakt, že pro spuštění aplikace v JavaScriptu, vystačí pouze prohlížeč s podporou této technologie, kdežto pro Java applety je nutná instalace JDK na uživatelský počítač. Mezi další přednosti JavaScriptu patří standardní povolení této technologie přímo v prohlížeči. JavaScript je většinou v nově nainstalovaném prohlížeči primárně povolen, kdežto Java applet se pokaždé dotazuje, zda má být spuštěn.

Aplikace, které zde byly vytvořeny, jsou funkční ve všech prohlížečích. U JavaScriptu je zobrazení lehce odlišné, protože každý prohlížeč využívá vlastní řešení podpory JavaScriptu. Oproti tomu Java applet dodržuje svůj vzhled, jelikož se spouští přes JDK, který má uživatel nainstalovaný v počítači.

Výsledné porovnání výrazně neupřednostňuje ani jednu z technologií. Naopak je znatelná určitá vyrovnanost v načítání obou aplikací. Výjimku tvoří načítání Java appletů v prohlížeči Safari u první aplikace, kde načtení Java appletu trvalo několikanásobně déle než v konkurenčních prohlížečích. Obecně má Java applet první načtení časově náročnější, jelikož se stahuje do prohlížeče a tím prodlužuje čas načtení stránky. Pokud se stránka

načítá opětovně je doba zkrácena na minimum. JavaScript má tento proces odlišný, protože samotný skript je součástí HTML a je i tak načítán, a proto pokaždé nabíhá znovu, ale přesto je rychle zobrazen.

Z analýzy vyplynulo, že prohlížeče Internet Explorer, Firefox a Safari rychleji načetly JavaScript než Java applet u první aplikace. Tyto prohlížeče představují 79% zastoupení na současném tržním podílu internetových prohlížečů. U druhé aplikace se naopak prokázala výhoda v rychlosti načítání Java appletu, který je pro složitější aplikace vhodnější.

8. Seznam použitých zdrojů

[1] BOLLINGER, Gary, BHARATHI, Natarajan. JSP - Java Server Pages (podrobný průvodce začínajícího tvůrce). 1. Vydání. Vystaveno Grada Publishing a.s. Praha 2003. 420 s. ISBN: 80-247-0340-8.

[2] ČADA, Ondřej. Objektové programování (naučte se pravidla objektového myšlení). 1. Vydání. Vystaveno Grada Publishing a.s. Praha 2009. 200 s. ISBN: 978-80-247-2745-5.

[3] FERSCHMANN, Petr. Bezpečnost na webu – přehled útoků na webové aplikace. [online]. Publikováno 10.11.2008 [cit. 2011-09-03]. Dostupné z: http://zdrojak.root.cz/clanky/prehled-utoku-na-webove-aplikace/.

[4] GRIMMICH, Šimon. Tvorba webu [online]. Vystaveno Copyright © 2003 – 2008. Dostupné z: http://www.tvorba-webu.cz/.

[5] HLAVENKA, Jiří. Vytváříme www stránky a spravujeme moderní web site. 7. Aktualizované vydání. Vystaveno CP Books, a.s. Brno 2005. 349 s. ISBN: 80-251-0801-5.

[6] JANOVSKÝ, Dušan. Jak psát web [online]. Vystaveno 2004. Dostupné z: http://www.jakpsatweb.cz/.

[7] KUČERA, František. Protokol HTTP [online]. Publikováno 12.9.2011 [cit. 2011-09-02]. Dostupné z: http://zdrojak.root.cz/clanky/protokol-http/.

[8] PROSISE, Jeff. Programování v Microsoft .NET. 1. Vydání. Vystaveno Computer Press. Brno 2003. 712 s. ISBN: 80-7226-879-1.

[9] ŠKULTÉTY, Rastislav. JavaScript (Programujeme internetové aplikace) 2. Aktualizované vydání. Vystaveno Computer Press 2004.224s. ISBN: 80-251-0144-4.

[10] SEMECKÝ, Jiří. Interval: Webdesign a e-komerce [online]. Vystaveno Zoner software, s.r.o. 26.4.2002. ISSN 1212-8651. Dostupné z: http://interval.cz/clanky/naucte-se-javu-uvod/.

[11] MORKES, David. Oživování www stránek pomocí skriptů. 1. Vydání. Vystaveno Grada Publishing a.s. Praha 2002. 192 s. ISBN: 80-247-0325-4.

[12] KODÝTEK, Pavel. Historie internet [online]. Publikováno 31.1.2006 [cit. 2011-11-16]. Dostupné z: http://www.webdesign.paysoft.cz/clanky/2006/historie-internetu/.

[13] AUJEZDSKÝ, Josef. Forma počítačového programu. [online]. [cit. 2011-11-18]. Dostupné z: http://www.root.cz/specialy/licence/forma-pocitacoveho-programu/.

[14] ZAKHOUR, Sharon a kolektiv. Java 6 výukový kurz. 1. Vydání. Vystaveno Computer Press 2007. 534 s. ISBN: 978-80-251-1575-6

[15] DEEPAMANOHAR. Calculating Page Loading Time. [online]. Publikováno 23.4.2008. Dostupné z: http://www.dreamincode.net/code/snippet1908.htm.

[16] STATCOUNTER. Global Stats. [online]. Publikováno 2011. [cit. 2011-11-24]. Dostupné z : http://gs.statcounter.com/

[17] Script. In Iwebtool: Computer glossary. [online]. [cit. 2011-11-24]. iWEBTOOL 2005-08. Dostupny z: http://www.iwebtool.com/what_is_script.html.

9. Přílohy

9.1 Zdrojový kód kalkulačky v JavaScriptu

```
x = 0;
ops = "n";
token = 0;
function calc(op)
{ if(!isNaN(op) || op==".")
  { if(!token)
    { if(document.calculator.win.value == 0)
      { document.calculator.win.value = op; }
      else
      { document.calculator.win.value = document.calculator.win.value + op; }    }
    else
    { document.calculator.win.value = op;
      token = 0;  }
    return;  }
  else
  { if(op=="C")
    { document.calculator.win.value = 0;
      token = 0;
      return;    }
     if(op=="CE")
    { document.calculator.win.value = 0;
      return;   }
        if(op=="%")
    { document.calculator.win.value = document.calculator.win.value / 100.0;
      token = 1;
      return;    }
    if(op=="+/-")
```

```
{ document.calculator.win.value = -document.calculator.win.value;
  token = 1;
  return;    }
if(op=="+" || op=="*" || op=="/" || op=="-" || op=="=")
{ token = 1;
  if(ops!="n")
  { if(ops=="+")
    { x = x -(- document.calculator.win.value);
      document.calculator.win.value = x;        }
    if(ops=="-")
    { x = x - document.calculator.win.value;
      document.calculator.win.value = x;    }
    if(ops=="/")
    { x = x / document.calculator.win.value;
      document.calculator.win.value = x; }
    if(ops=="*")
    { x = x * document.calculator.win.value;
      document.calculator.win.value = x;  }      }
  else
  { x = document.calculator.win.value; }
if(op!="=") { ops=op; }
  else { ops="n"; }
  return;   }  }}
```

Simple Javascript Calculator. *Zdroj: http://www.scripts.com/javascript-scripts/calculator-scripts/simple-javascript-calculator/.*

9.2 Zdrojový kód kalkulačky v Java appletu

```
import java.awt.*;
import java.awt.event.*;
```

```
public class calculator extends java.applet.Applet implements ActionListener {
        TextField txtTotal = new TextField("");
   Button button[] = new Button[10];
        Button divide = new Button("/");
        Button mult = new Button("*");
        Button plus = new Button ("+");
        Button minus = new Button("-");
        Button isequalto = new Button("=");
        Button clear = new Button("CA");
        double num ,numtemp ;
        int counter;
        String strnum = "",strnumtemp = "" ;
        String op = "";
public void operation() {
         counter ++;
                  if (counter == 1) {
                   numtemp = num;
                   strnum = "";
                   num = 0;
                  }else{
                   if (op == "+") numtemp += num;
                   else if (op == "-") numtemp -= num;
                   else if (op == "*") numtemp = numtemp * num;
                   else if (op == "/") numtemp = numtemp / num;
                   strnumtemp = Double.toString(numtemp);
                   txtTotal.setText(strnumtemp);
                   strnum = "";
```

```
                    num = 0; }}
public void init() {
 setLayout(null);
 plus.setBackground(Color.blue);
          plus.setForeground(Color.white);
    minus.setBackground(Color.blue);
          minus.setForeground(Color.white);
    divide.setBackground(Color.blue);
          divide.setForeground(Color.white);
          isequalto.setBackground(Color.blue);
          isequalto.setForeground(Color.white);
          mult.setBackground(Color.blue);
          mult.setForeground(Color.white);
          clear.setBackground(Color.blue);
          clear.setForeground(Color.red);
 for(int i = 0;i <= 9; i ++) {
                    button[i] = new Button(String.valueOf(i));
                    button[i].setBackground(Color.orange);
                    button[i].setForeground(Color.blue); }
button[1].setBounds(0,53,67,53);
          button[2].setBounds(67,53,67,53);
          button[3].setBounds(134,53,67,53);
          button[4].setBounds(0,106,67,53);
          button[5].setBounds(67,106,67,53);
          button[6].setBounds(134,106,67,53);
          button[7].setBounds(0,159,67,53);
          button[8].setBounds(67,159,67,53);
          button[9].setBounds(134,159,67,53);
```

```
 for (int i = 1;i <= 9; i ++) {
                    add(button[i]); }
 txtTotal.setBounds(0,0,200,53);
          add(txtTotal);
          plus.setBounds(0,212,67,53);
          add(plus);
          button[0].setBounds(67,212,67,53);
          add(button[0]);
          minus.setBounds(134,212,67,53);
          add(minus);
          divide.setBounds(134,264,67,53);
          add(divide);
          isequalto.setBounds(67,264,67,53);
          add(isequalto);
          mult.setBounds(0,264,67,53);
          add(mult);
          add(clear);}
public void start() {
          for(int i = 0;i <= 9; i ++) {
                    button[i].addActionListener(this); }
          plus.addActionListener(this);
          minus.addActionListener(this);
          divide.addActionListener(this);
          mult.addActionListener(this);
          isequalto.addActionListener(this);
          clear.addActionListener(this);}
public void stop() {
          for(int i = 0;i <= 9; i ++) {
```

```
                button[i].addActionListener(null); }
        plus.addActionListener(null);
        minus.addActionListener(null);
        divide.addActionListener(null);
        mult.addActionListener(null);
        isequalto.addActionListener(null);
        clear.addActionListener(null);}
public void actionPerformed(ActionEvent e) {
                for(int i = 0;i <= 9; i++) {
                        if (e.getSource() == button[i]) {
                          play(getCodeBase(),i + ".au");
                          strnum += Integer.toString(i);
                          txtTotal.setText(strnum);
                          num = Double.valueOf(strnum).doubleValue();    }}
   if (e.getSource() == plus) {
                 operation();
                 op = "+";                }
                if (e.getSource() == minus) {
                 operation();
                 op = "-";                }
                if (e.getSource() == divide) {
                 operation();
                 op = "/";                }
                if (e.getSource() == mult) {
                 operation();
                 op = "*";                }
                if (e.getSource() == isequalto) {
                 if (op == "+") numtemp += num;
```

```
                else if (op == "-") numtemp -= num;
                else if (op == "*") numtemp = numtemp * num;
                else if (op == "/") numtemp = numtemp / num;
          strnumtemp = Double.toString(numtemp);
                txtTotal.setText(strnumtemp);
                strnumtemp = "";
                numtemp = 0;
                strnum = "";
                num = 0;
                counter = 0;              }
               if (e.getSource() == clear) {
                txtTotal.setText("0");
                strnumtemp = "";
                numtemp = 0;
                strnum = "";
                num = 0;
                counter = 0;              }  }  }
```

CalcApplet. *Zdroj: http://www.javafile.com/calc/adder/adder.php.*