Univerzita Komenského v Bratislave

Fakulta matematiky, fyziky a informatiky

Wikinavigátor

Bratislava, 2011

Martin Bielik

Univerzita Komenského v Bratislave

Fakulta matematiky, fyziky a informatiky

Katedra informatiky

Wikinavigátor

(Bakalárska práca)

Študijný program:	Informatika
Študijný odbor:	9.2.1 Informatika
Školiace pracovisko:	Katedra algebry, geometrie a didaktiky matematiky
Vedúci:	RNDr. Jana Katreniaková
Kód:	722bce37-29a8-41

Bratislava, 2011

Martin Bielik

Čestne prehlasujem, že bakalársku prácu som vypracoval samostatne s použitím uvedenej literatúry a pod dohľadom mojej vedúcej práce.

. . . . . .

Chcel by som poďakovať mojej vedúcej RNDr. Jane Katreniakovej za jej veľkú pomoc a za dohľad nad mojou činnosťou.

Abstrakt

Internetové prehliadače sa neustále vyvýjajú. V dnešnej dobe existuje niekoľko prehliadačov, ktoré používajú ľudia na celom svete. Ich vývojári sa snažia naprogramovať čo najpríjemnejšie, dobre fungujúce a zároveň jednoduché a userfriendly prostredie, aby prilákali čo najviac používateľov.

Niekedy však základná funkcionalita webovského prehliadača nestačí. Každému používateľovi sa môže páčiť niečo iné, alebo potrebuje nejakú inú podporu pre browsovanie a prácu s internetom. Nedá sa však vyhovieť všetkým a preto existujú do prehliadačov rozšírenia. Sú to drobné programy ktoré môžu doplniť funkcionalitu prehliadača a každý používateľ si môže vybrať, ktoré sa mu páčia a ktoré bude používať.

Mojou úlohou v tejto práci, je naprogramovať rozšírenie Wikinavigátor do prehliadača Mozilla Firefox. Toto rozšírenie by malo uľahčiť používateľom orientáciu vo svojej histórii svojho browsovania. Tá bude zobrazená ako graf, ktorého bunku budú navštívené www stránky a hrany budú prechody medzi nimi.

KĽÚČOVÉ SLOVÁ: Wikinavigátor, Mozilla Firefox, Doplnky, Rozšírenia

Abstract

Internet browsers are still evolving. Nowadays there are some browsers which are used all over the world. Developers are trying to program confortable, well working but also simple and user-friendly enviroment to attract more users.

But sometimes basic functionality is not enought. Each user likes different things and he usually needs some special functionality to do with the internet. Browser can not contain everything and therefore, there are an extensions for browsers. These are small programs which can add some new functionality and each user can choose which one he wants.

My task is to program Firefox extension called Wikinavigator. This extension should make users orientation in browser history easier. History will be displayed in graph, which nodes are visited pages and relations are conjunction between them.

KEYWORDS: Wikinavigator, Mozilla Firefox, Extensions, Add-on

Zoznam obrázkov

Obsah

Úvod

V dnešnej dobe sa stal internet veľmi dôležitou súčasťou života mnohých ľudí. Bežný užívateľ k nemu pristupuje pomocou webovského prehliadača a browsuje na stránkach. Existuje viacero prehliadačov a je na užívateľoch ktorý z nich si vyberú, ktorý ich upúta svojim vzhľadom či funkcionalitou. Preto sa každý prehliadač snaží byť ten najlepší, upútať čo najviac používateľov. Takáto konkurencia a súťaž je dobrá, pretože developori sa snažia prísť vždy s niečím novým a lepším. Do prehladačov existujú doplnky, ktoré vytvárajú developeri. Ich úlohou je spríjemniť, uľahčiť, urýchliť prácu s internetom. Užívateľia si potom vyberú ktorý doplnok sa im pre svoj prehliadač páči a môžu si ho nainštalovať.

Mojou úlohou je vytvoriť doplnok Wikinavigátor do prehliadača Mozilla Firefox. Jeho úlohou je zjednodušiť orientáciu na stránkach, ktoré užívateľ už navštívil od začiatku jeho browsovania. Tento plugin zobrazí v novom tabe prehliadača všetky stránky ktoré navštívil a ako sa k nim dopracoval.

Prácu som rozdelil na päť kapitol.V prvej kapitolo popíšem prehliadač Mozilla Firefox, aké typy doplnkov do neho sa tvoria, čo treba vedieť o tých doplnkoch a aké technológie sa pri tvorbe používajú.

V druhej kapitole špecifikujem presne čo má tento doplnok presne robiť, ako bude vyzerať v prehliadači a ako sa bude používať.

V tretej kapitole popíšem akým spôsobom vykreslujem a zobrazujem do prehliadača navštívené stránky, prečo to robím práve takto a ako by sa to ešte inak dalo

Štvrtá kapitola bude obsahovať nejaké zaujímavé veci ktoré som použil pri vytváraní projektu, popis ako tento program funguje a ukážky z kódu.

Kapitola 1

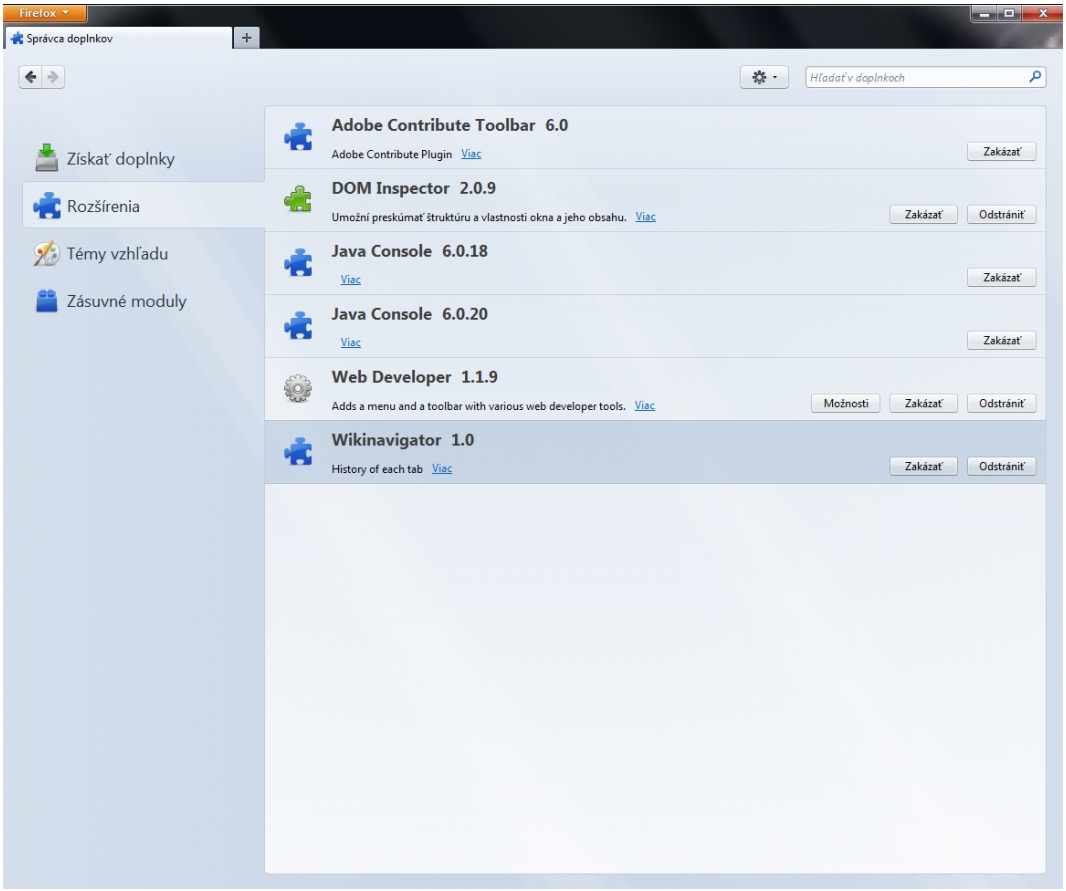
Mozilla Firefox - Rozšírenia

1.1 Mozilla Firefox

Mozilla Firefox je voľný a open source webový browser, ktorý vyvýja Mozilla Foundation. Z posledných štatistík z apríla 2011 je Firefox s takmer 43 percentnou používanosťou najpoužívanejším prehliadačom webových stránok na svete. Na ďaľších miestach je Google Chrome (25%), Internet Explorer (23%) a Safari (4%). Na zobrazovanie stránok používa Firefox Gecko layout engine ktorý implementuje najpoužívanejšie webové štandardy. Firefox je Cross-platformový program, beží na najviac používanejších operačných systémoch ako Microsoft Windows, Mac OS, Linux a mnohých ďaľších. Momentálne je najnovšia verzia Firefox 4.0 ktorá vyšla začiatkom roku 2011. Podporuje webové štandardy ako HTML5, CSS3, technológiu Web GL (javascript API pre zobrazovanie 3D grafiky).

1.2 Doplnky

Do Firefoxu je možné doinštalovať si mnohé **doplnky** (add-ones), ktoré používaťeľom uľahčujú, spríjemňujú a zrýchľujú browsovanie na internete. Doplnky svojho Firefoxu môžeme vidieť pod možnosti - doplnky (obrázok 1.1). Väčšina doplnkov je sprístupnená na oficiálnej stránke: *https://addons.mozilla.org*



Obr. 1.1: Okno Firefoxu pre správu doplnkov

Doplnky sa delia na tri základné skupiny:

- **Rozšírenia** - takýto typ doplnku je Wikinavigátor
- **Témy vzhľadu**
- **Zásuvné moduly**

1.2.1 Rozšírenia

Rozšírenia (Extensions) sú malé programy ktoré si používateľ môže nainštalovať do Firefoxu. Ich úlohou zvyčajne býva modifikovať, alebo pridať nejaké nové vlastnosti prehliadača tak, aby používateľom uľahčili prehliadanie, prácu a aby im spríjemnili toto webovské prostredie.

Rozšírenia sa používajú napríklad pre správu záložiek, RSS, FTP účtov (napr.: FireFTP), História a mnohých iných častí Firefroxu, uľahčenie práce pomocou myšky(All-in-One-gestures), atď.

Veľký význam majú pre web developerov ktorým uľahčujú kopu práce. Také doplnky sú napríklad Web Developer, DOM Inspector, FireBug, atď.

Ďalej sa požívajú na modifikovanie webstránok, ktoré si používateľ otvorí. Napr. pre pridanie nejakých možností na google stránky, atď.

Veľmi zaujímavý doplnok pre modifikovanie stránok je GreaseMonkey, ktorý umožnuje písať používateľom skripty v javascripte pre vybrané stránky, povoľuje k nim pristupovať cez DOM a DHTML(Dynamic HTML). Tým pádom píše používateľ určitý typ doplnku bez znalosti vytvárania doplnkov do Firefoxu.

Pre vytvorenie takéhoto doplnku sa používajú niektoré webovské technológie, napríklad:

- **JavaScript** - základný programovací jazyk pre Firefox
- **CSS** - kaskádové štýly
- **DOM** - (Document Object Model) používa sa na menenie, a správu XML(v XUL alebo HTML) buniek v reálnom čase.

Používajú sa však aj technológie ktoré vynašla a používa Mozilla Foundation, napríklad:

- **XUL** - na základe XML popisuje interface Firefoxu (všetky Buttony, Menu, StatusBary, atď.)
- **XPCOM** - Cross-Platform Component Object Model
- **XPI** - Cross-Platform Installer

1.2.2 Témy vzhľadu

Témy do firefoxu sú inštalačné súbory, v ktorých tvorca popisuje ako majú jednotlivé časti firefoxu vyzerať. Hlavnou úlohou je spríjemniť používateľovi prostredie podľa vlastného výberu témy.

1.2.3 Zásuvné moduly

Pluginy sú doplnky ktoré rozširujú možnosti Firefoxu. Napríklad pre prehrávanie videa, skenovanie pred vírusmi, pre zobrazovanie nových typov súborov.
Vo Firefoxe sú veľmi známe pluginy pre Adobe Flash Player, QuickTime, Adobe Acrobat, atď.

Kapitola 2

Špecifikácia projektu

2.1 Popis projektu

Wikinavigátor je doplnok do Mozilla Firefoxu, ktorý zobrazuje históriu browsovania v jednotlivých taboch Firefoxu. Používateľ by sa mal ľahko dozvedieť ako sa dopracoval k rôznym stránkam, z kade na ktorú prišiel a mal by sa vedieť k všetkým stránkam, na ktorých browsoval, dostať.

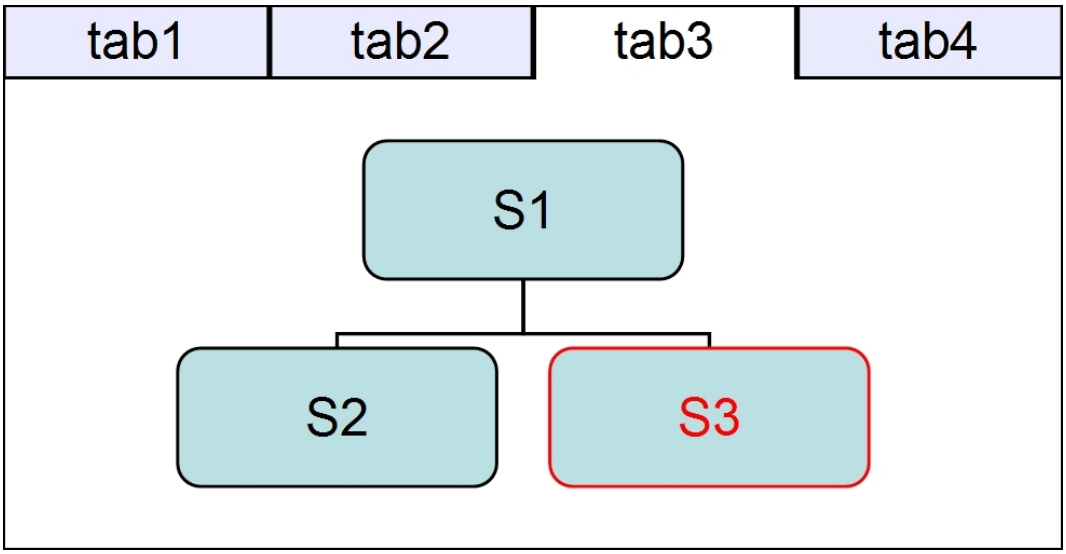
2.2 Funkcionalita a dizajn

Podrobný popis projektu môžem zhrnúť do niekoľkých bodov:

- Po nainštalovaní a reštarte Firefoxu sa zobrazí v browseri tlačidlo Wikinavigátoru s logom (vpravo hore)
- Po kliknutí na Tlačidlo sa otvorí nový tab, v ktorom budú zobrazené všetky ostatné taby, bude sa medzi nimi dať prepínať a focus bude mať naposledy použitý tab
- V týchto taboch budú aj taby, ktoré už používateľ zatvoril (budú inak zvýraznené)
- Pre zvolený tab vykreslí Wikinavigátor jeho históriu v podobe stromu
- Bunky stromu budu zmenšené screenshoty stránok ktoré používateľ navštívil, pri prechode sa zväčšia používateľ uvidí aj základné informácie o stránke ako

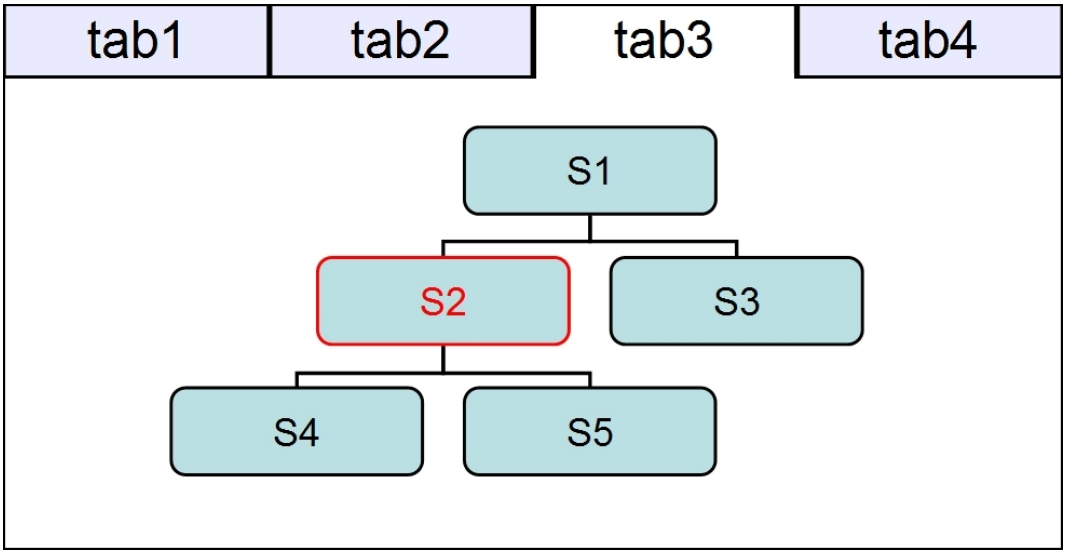
title a URL (každá bunka si pamätá screenshot, URL, title stránky okrem iných vecí)

- Aktuálna bunka bude zvýraznená
- Po kliknutí na bunku sa bunka stane aktuálnou, focus dostane tab ktorému prislúcha a do tabu sa načíta adresa bunky
- Vyťahy medzi bunkami budú vykreslené podľa toho ako používateľ browsoval (tak ako je to zobrazené na obrázkoch 2.1 a 2.2)
- Po opakovanom kliknutí na tlačidlo Wikinavigátoru sa buď prenesie focus na tab s Wikinavigátorm a ak taký nieje, tak sa otvorí nový tab so všetkými dátami

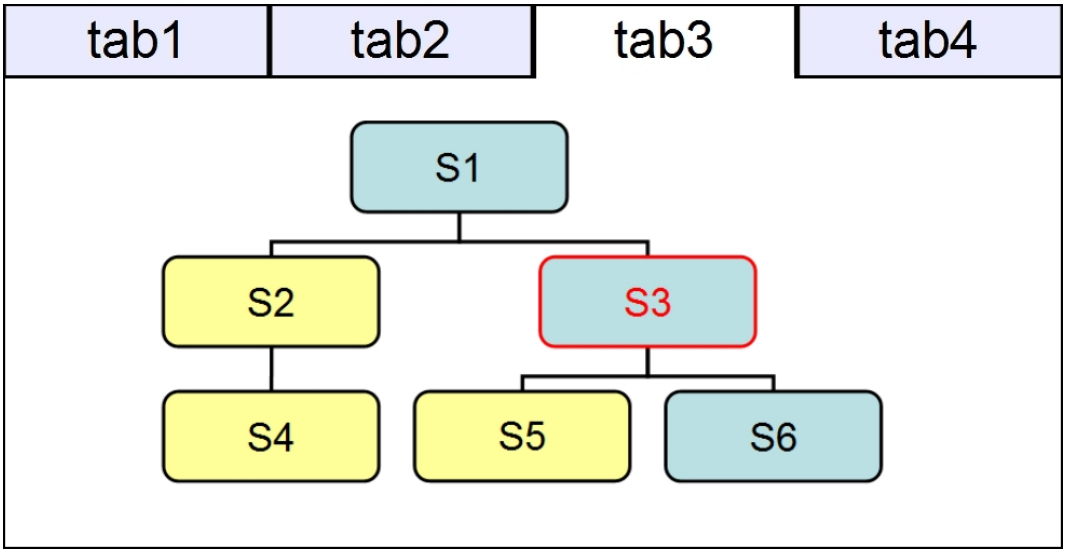


Obr. 2.1: Browsovanie v podobe stromu - Používateľ v tabe3 pravdepodnobne browsoval takto: najprv navštívil stránku S1, potom S2, potom sa vrátil na stránku S1 a navštívil stránku S3

Okrem prehľadnosti browsovania je veľkou výhodou Wikinavigátoru, že používateľ môže vidieť aj stránky ktoré navštívil a už sa k nim nedá dopracovať pomocou tlačidiel History Back a History Forward. Vo vykreslenom strome to nazvem mŕtve časti histórie ako je vidieť na obrázku 2.3.



Obr. 2.2: Browsovanie v podobe stromu - Používateľ mohol v tabe3 browsovať takto: S1, S2, History Back, S3, History Back, S2, S4, History Back, S5, History Back



Obr. 2.3: Mŕtve časti histórie - Aktívna stránka a bunka je S3, používateľ sa môže pomocou History Back a History Forward dostať k stránkam S1 a S6. Žlto označené bunky sú mŕtve časti histórie a používateľ sa k nim môže dopracovať pomocou Wiki-navigátoru

Kapitola 3

Vykresľovanie

3.1 Vykresľovanie stromu Wikinavigátoru

Na vykresľovanie stromu navštívených stránok som si zvolil vlastný algoritmus, ktorý má výhody a aj nevýhody, ktoré popíšem neskôr. Každý strom histórie reprezentuje trieda Navigator, ktorá má inštancie a metódy ako na obrázku 2.4. Každú bunku stromu reprezentuje trieda Node, ktorá má inštancie a metódy ako na obrázku 2.5. Triedy sa nachádzajú v Namespacei Wikinavigator.Classes. Inštanciea triedy navigator Root a Active sú typu Node. Pole Childs v triede Node je pole detí príslušnej bunky. Práve cez inštanciu Root, jej metódami hasChild, hasParent a poľom Childs pristupujem k celému stromu.Hlavná metóda na vykresľovanie je DrawNodes, ktorá pre každú bunku zavolá metódu Draw v triede Node.

```
1  Wikinavigator.Classes.Navigator = function(){
2
3      this.ID; // id tabu
4      this.Root;
5      this.Active;
6      this.MaxDepth;
7         ...
8
9      this.Draw = function(b){} // vykreslí bunky
10     this.AddNode = function(){} // pridá bunku
11        ...
12
13 }
```

Obr. 3.1: Trieda Navigator

```
1 Wikinavigator.Classes.Node = function(){
2
3     this.X;
4     this.Y;
5     this.URI;
6     this.Childs = new Array();
7     this.ImageData;
8     this.ID; // id bunky
9     this.Title;
10    this.Depth;
11       ...
12
13    this.AddChild = function(){} // pridá dieťa
14    this.Draw = function(x, y){} // vykreslí bunku
15    this.hasParent = function(){}
16    this.hasChild = function(){}
17       ...
18
19 }
```

Obr. 3.2: Trieda Node

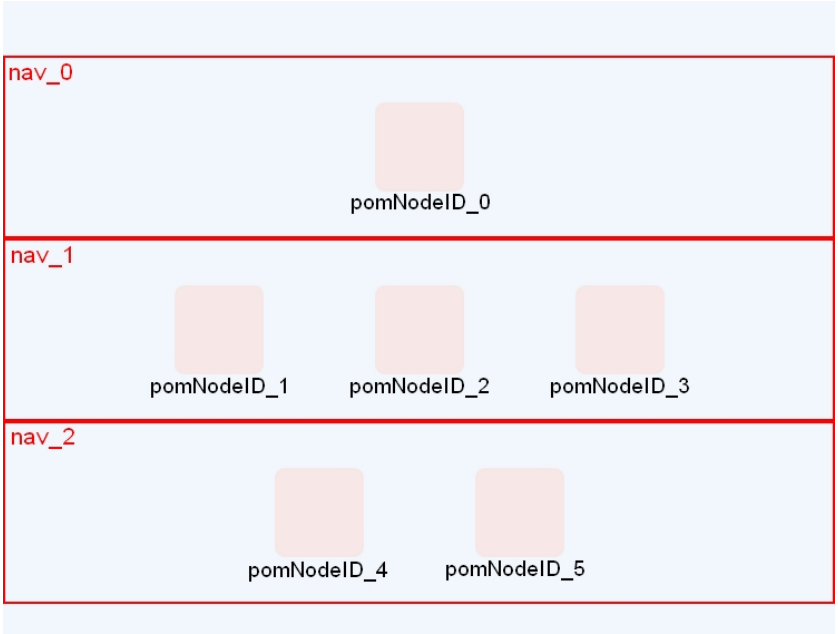
Ako som už spomínal, po kliknutí na tlačidlo Wikinavigátoru sa otvorí nový tab v ktorom budú zobrazené všetky ostatné taby a bude vykreslený storm histórie aktívneho tabu. Okno v ktorom to vidíme, je popísané v HTML súbore index.html. Ten sa vytvorí po inštalácii doplnku. Základná štruktúra tohoto súboru je zobrazená na obrázku 2.6. Hlavný je blok s id "main", pod ním je list ul s id "navigation"v ktorom sa zobrazuje navigácia tabov Firefoxu. V bloku s id "navigator"sa vykresľuje strom histórie. Pri zavolaní funkcie na vykreslenie stromu do neho hlavný script Wikinavigátoru bude zapisovať pomocou DOMu (Document Object Model) rôzne divy, vykreslí všetky bunky daného stromu a nakreslí vzťahy medzi bunkami.

```
1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
2 <html>
3   <head>
4     <title>Wikinavigator</title>
5   </head>
6   <body>
7     <div id="main">
8       <ul id="navigation"></ul>
9
10      <div id="navigator">
11      </div>
12
13    </div>
14  </body>
15 </html>
```

Obr. 3.3: index.html, strínka na ktorej sa vykresľuje strom histórie

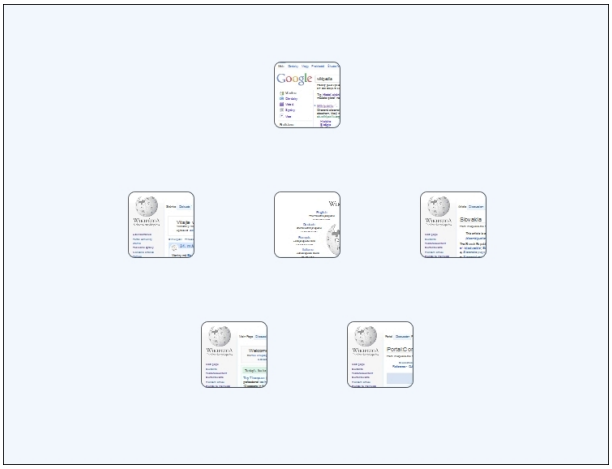
Postup ako sa vykresľuje strom histórie tabu:

- V triede Navigator sa zavolá metóda DrawNodes().
- V nej sa pre každú hĺbku stromu sa pridá div s id "nav_"+ hĺbka ,napríklad nav_0, nav_1, ... (vlastnosť MaxDepth).
- Text v bloku "navigator"sa centruje na stred
- Pre každú bunku teraz vloží pomocný div s id "pomNodeID_"+ ID bunky do divu "nav_"+ príslušná hĺbka ako vidíme na obrázku 3.4.



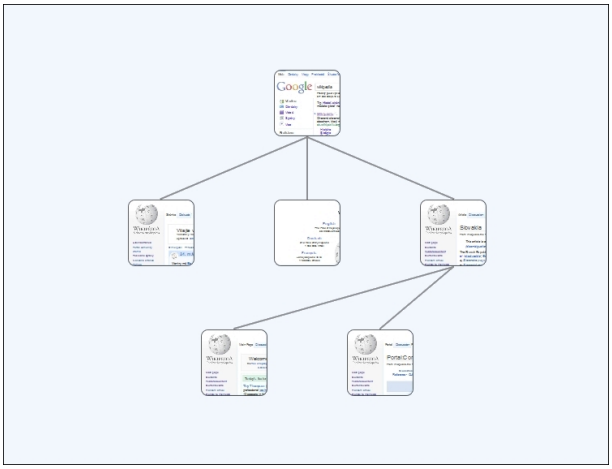
Obr. 3.4: bloky umiestnené v bloku "navigator", prvá fáza vykresľovania. Všetky tieto zvýraznené bloky sú len pomocné pri algoritme vykresľovania aby určili polohu buniek stromu. Vo Wikinavigátore ich nieje vidno.

- Pre každú bunku je zavolaná funkcia Draw s parametrami x a y ako je offsetTop a offsetLeft príslušného pomocného divu ("nav_"+ ID bunky).
- Funkcia Draw vloží div s id "divPrevID_"+ ID bunky ktorý obsahuje príslušné informácie o bunke. Tomuto divu nastaví CSS vlastnosť position na äbsoluteä umiestni ho na dané súradnice.



Obr. 3.5: Na pomocné bloky boli pripnuté bloky "divPrevID_"+ ID bunky. To sú divy ktoré reprezentujú bunku stránky.

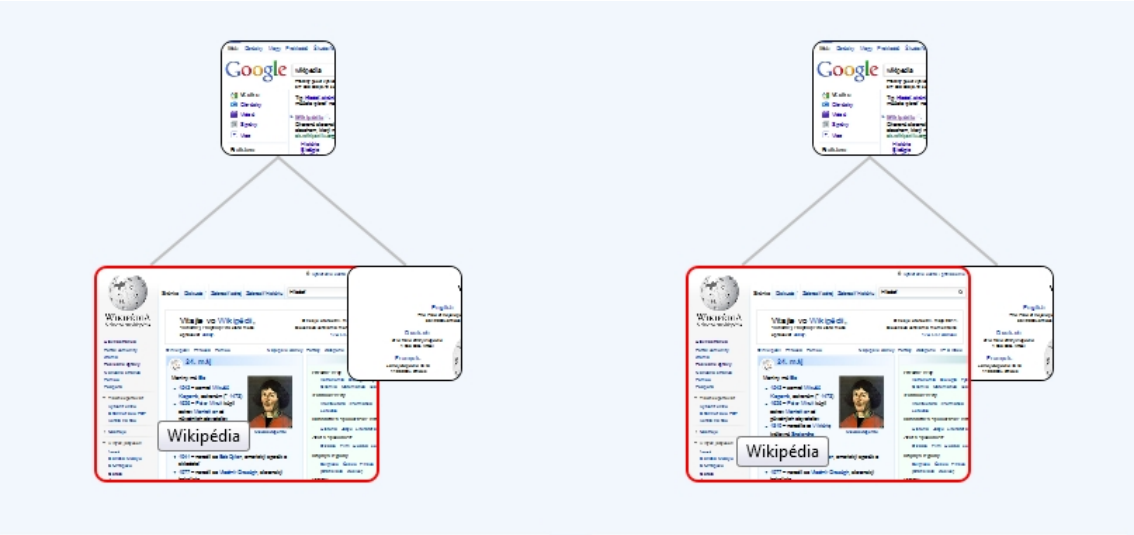
- Ako posledný krok prejde všetky bunky a vykreslí vzťahy medzi nimi.



Obr. 3.6: Výsledný strom z vykreslenými vzťahmi.

Na vykresľovanie buniek teda využívam HTML vlastnosti divov, ich centrovanie, floatovanie a zarovnania. Pomocné bloky potrebujem na to aby som si ich rozhodil do prísluáných hĺbkových blokov. Keby tam vkladám priamo bloky ktoré reprezentujú bunku ("divPrevID_"+ ID bunky) tak by mohlo nastať zlé prekrytie pri náhľade bloku ako je znázornené na obrázku 3.7. Keď bunky pripnem na pomocné bloky v

správnom poradí, tak je prekrývanie v poriadku.



Obr. 3.7: Po prejdení myškou ponad bunku sa bunka zväčší. Ak by som bunky priamo ukladal v poradí ako vkladám pomocné bunky nastane nesprávny efekt (vľavo), správný je druhý obrázok (vpravo)

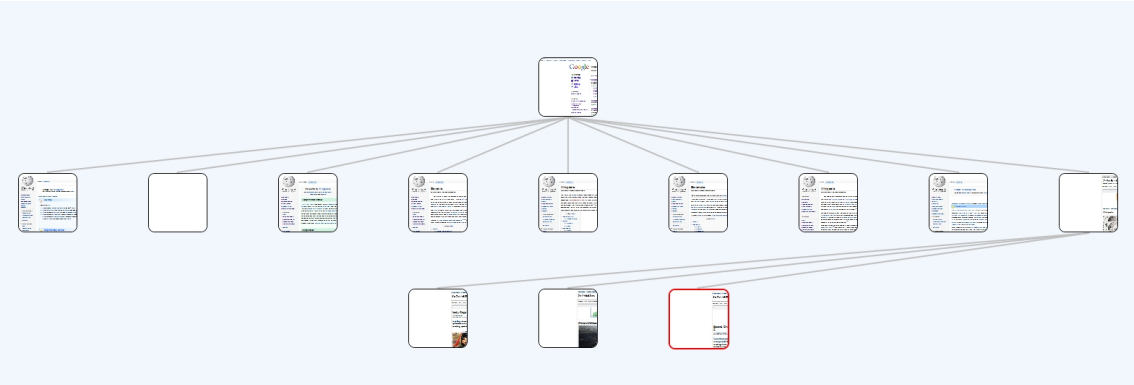
Výhody vykresľovacieho algoritmu:

- Tým že využívam HTML vlastnosti blokových elementov, nezaťažujem program počítaním X-ových a Y-ových súradníc každej bunky. Tým pádom je tento algoritmus rýchly. Na to aby sa strom vykreslil, stačí aby prešiel strom dva krát (prvý pri rozhodení pomocných blokov a druhý pri vykreslení blokov buniek).
- Do programu sa dajú ľahko implementovať aj iné algoritmy, ktoré by vykreslili strom. Stačilo by zmeniť funkciu DrawNodes() pre strom, v nej dopočítať súradnice buniek a vykresliť funkciou Draw(x,y) pre každú bunku.
- Keďže celý strom je centrovaný na stred, používateľ má bunky pokope. V prípade nejakých divokých grafov nemusí veľa používať horizontálny scrollbar Firefoxu aj keď sa to môže stať, ale nie až v takom rozsahu ako u iných algoritmov. (používanie horizontálneho scrollbaru je nevyhnutné ak používateľ browsoval dlho)

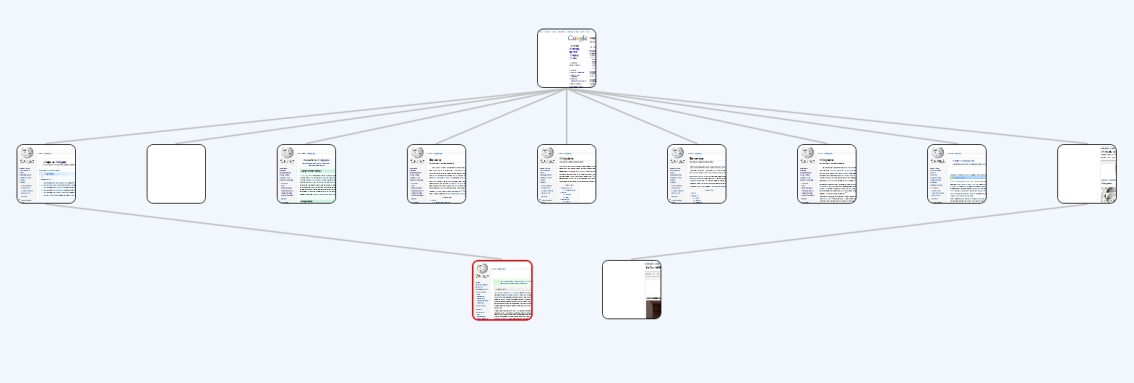
- Najvýhodnejší je, keď používateľ browsuje tak, že počte buniek v každej hĺbkovej úrovni niesú veľké roziely.

Nevýhody vykresľovacieho algoritmu:

- Pri rôznych divokých grafoch nevznikajú pekné, pravidelné vetvy stromu (napr.: obrázky 3.8 a 3.9).



Obr. 3.8: Príklad divokého stromu histórie pre môj algoritmus.

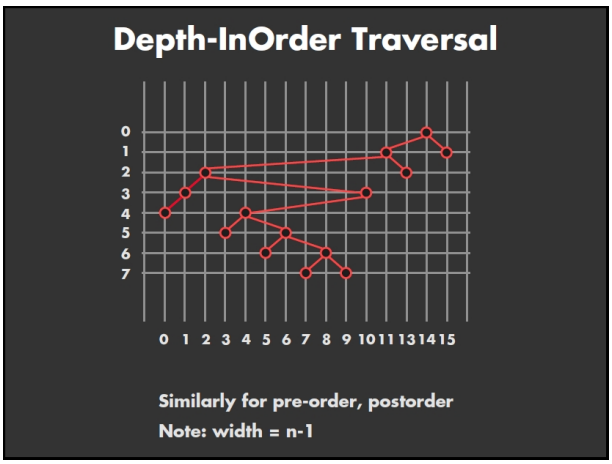


Obr. 3.9: Príklad divokého stromu histórie pre môj algoritmus.

3.2 Iné algoritmy na vykresľovanie stromov

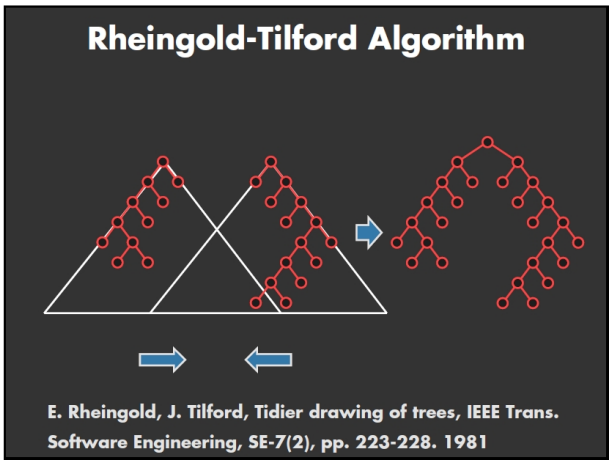
Ďalšie algoritmy na vykresľovanie stromov, ktoré by sa dali implementovať do Wiki-navigátoru, sú napríklad:

- **Depth-InOrder Traversal** algoritmus. Je to jednoduchý a rýchly algoritmus. Každá bunka je vo svojej hĺbke zobrazená v samostatnom stĺpci (obrázok 3.10). Vykreslený graf má pri veľkom počte buniek veľkú šírku, čo by bolo pri Wikinavigátore nevýhoda.



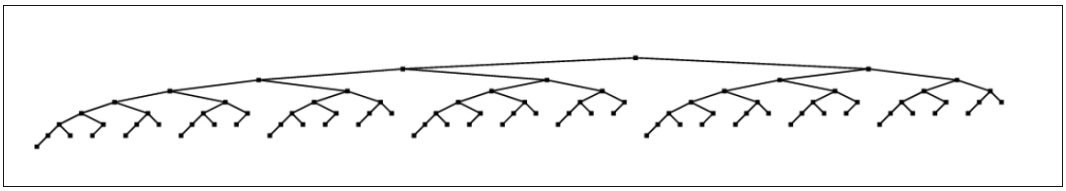
Obr. 3.10: Depth-InOrder Traversal Algorithm [20110a]

- **Rheingold-Tilford Algorithm** ako je na obrázku 3.11, ohraničuje od najspodnejších buniek trojuholníkové sekcie a tie potom spája. Pre Wikinavigátor môžu nastať tiež dosť široké grafy, ale je vhodnejší ako predchádzaúci algoritmus.



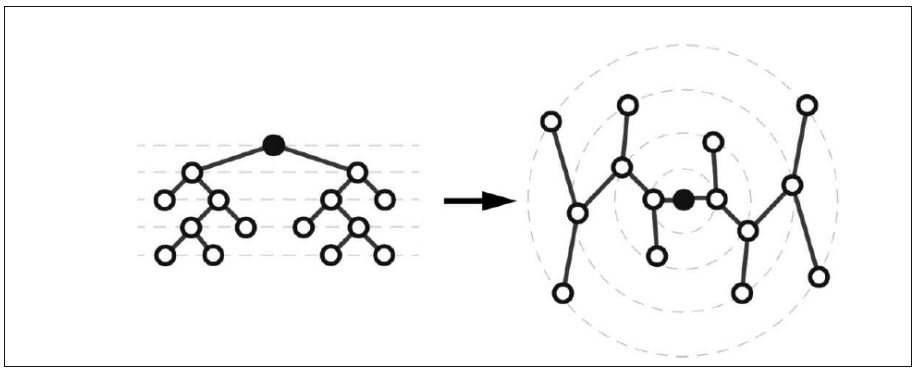
Obr. 3.11: Rheingold Tilford Algorithm [20110a]

- **Fibonačiho strom (E. Reingold and J. Tilford)**, obrázok 3.12, je tiež široký algoritmus.



Obr. 3.12: Drawing of the Fibonacci tree with 88 nodes [20001]

- **Radiačn algoritmy**. Strom sa zobrazuje na na kružniciach so stredom v začiatočnej bunke. Kružnica na ktorej je bunka závisí od jej hĺbky (obrázok 3.13).

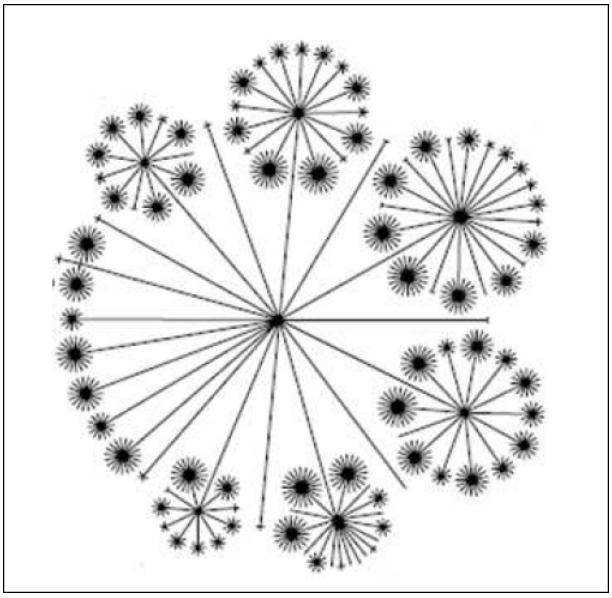


Obr. 3.13: Radial Drawing Algorithm [20001]

- **Cone trees (G. G. Robertson, J. D. Mackinlay, and S. K. Card)** (obrázok 3.14)
- Shiloach (1976), Crescenzi al. (1992), Chan al. (1996, 1999), Crescenzi and Piperno (1995) a iné

3.3 Vykresľovanie bunky

Pre vykreslenie bunky som objekt jazyka HTML5 - ¡canvas¿. Pomocou tohoto tagu sa dá ľahko pracovať s jednoduchou 2D grafikou ako kreslenie čiar, 2D objektov a aj

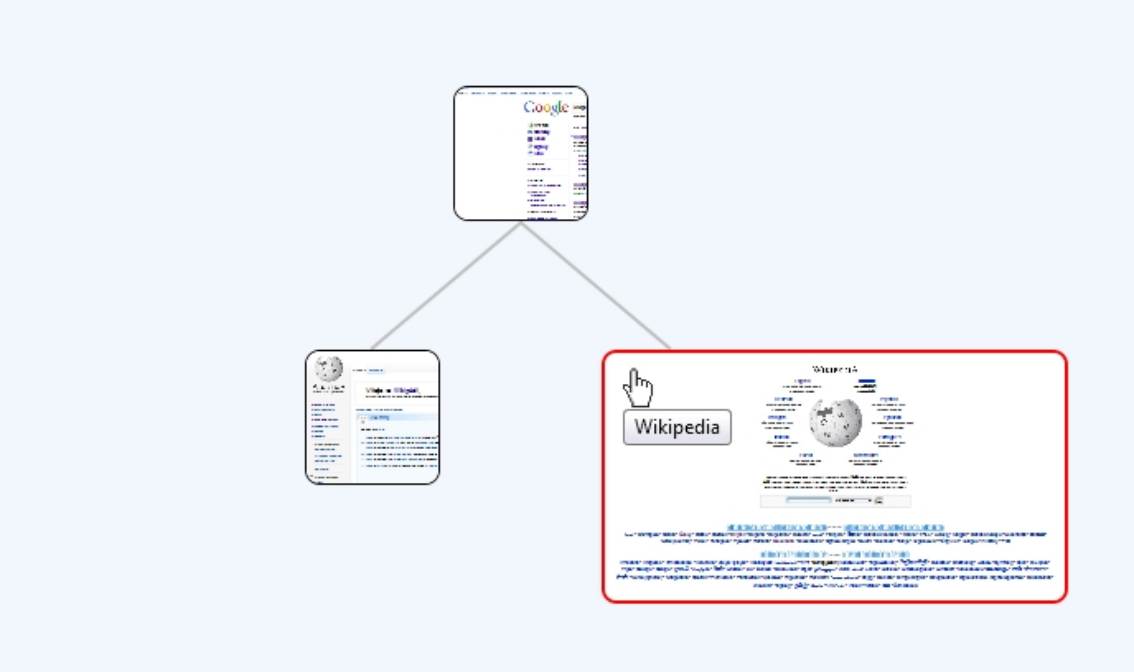


Obr. 3.14: Cone trees [20001]

obrázkov. Tento tag je vytvorený pre každú bunku. Pri načítaní stránky sa vytvorí nová bunka a do jej inštancie ImagData sa uložia dáta obrázku z tagu body. Pri vykresľovaní sa tieto dáta vykreslia na príslušnú canvas. V hlavnom dive bunky ("divPrevID_"+ ID bunky) sa normálne pri vykreslení zobrazí iba časť obrázku a pri prejdení myšou ponad bunku sa odokryje celý. Pri prejdení ponad bunku sa tiež zobrazí bublina (z atribútu divu title) v ktorej je title príslušnej stránky. Aktívna bunka je farebne zvýraznená (všetko na obr 3.10).

3.4 Vykresľovanie vzťahov medzi bunkami

Vykresľovanie vzťahov medzi bunkami znamená prejdenie celého stromu a pospájanie susedných buniek nejakou čiarou. Keďže bunky sa nevykresľujú na jedno plátno(canvas) nedá sa to pospájať jednoducho. Môj prvý nápad bol, fixne položiť nejaký dostatočne veľký canvas do pozadia stránky a na to vykresliť vzťahy. Takéto vrstvenie však v HTML5 nefunguje. Dalo by sa to spraviť aj tak, že by sa všetky bunky vykresľovali na jeden canvas spolu so vzťahmi. To však nefungovala vlastnosť hover(alebo aj onmousemove) na jednotlivých bunkách a naprogramovanie toho by bolo komplikované.

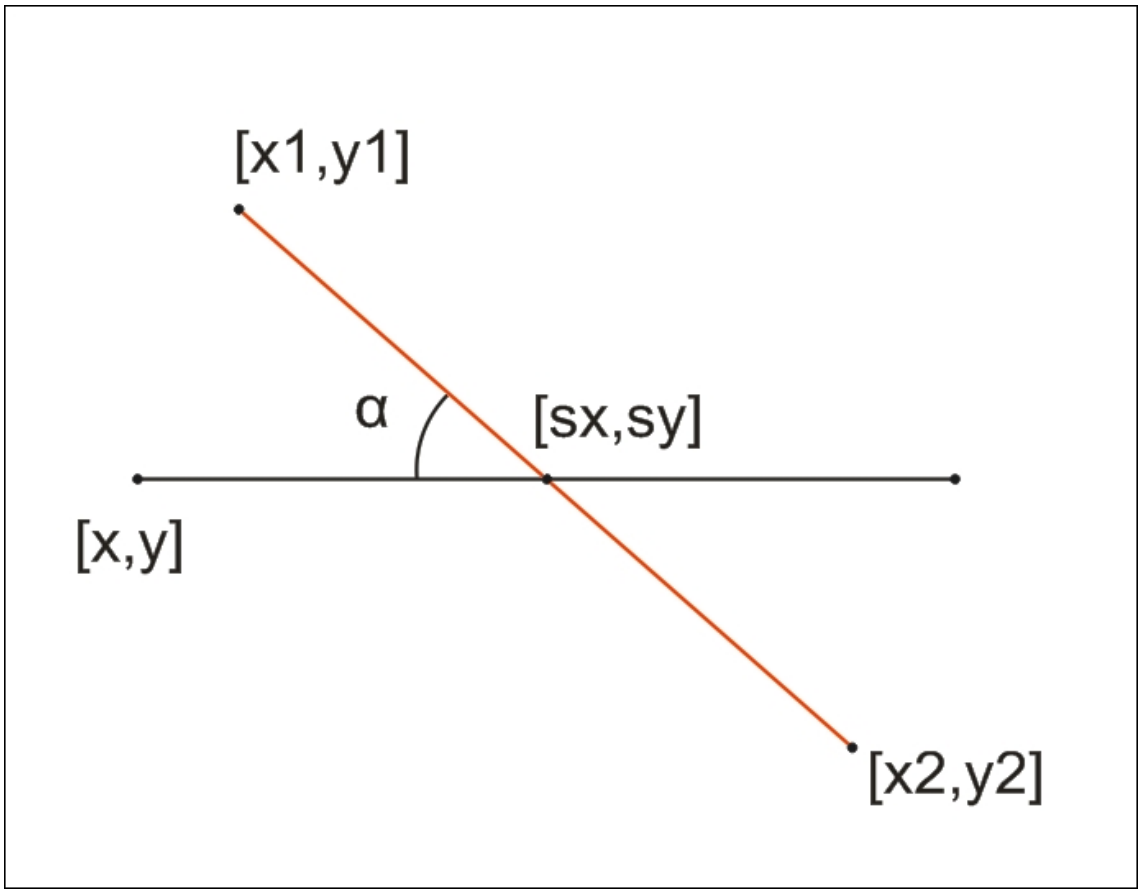


Obr. 3.15: Vykreslenie bunky. Červený rámček - aktívna, zároveň je nad ňou aj myška a title príslušnej stránky "Wikipedia".

Pre vykreslenie vzťahov som nakoniec používal naštýlované blokové elementy. Postupoval som takto:

- Do hlavného bloku som pridal nový s id "relations". V tomto bloku budú všetky elementy ktoré zobrazujú vzťahy.
- Pre každý vzťah medzi bunkami som si vypočítal x1, y1, x2, y2, čo sú súradnice čiary vzťahu
- Čiaru vykreslí privátna funkcia triedy Navigator DrawLine(x1, y1, x2, y2) a pri vykresľovaní postupuje následovne.
- Vytvorí nový blokový element(div) ktorému nastaví šírku akú má mať daná čiara a dĺžku 1 pixel.
- Nastavím rámček okolo bloku.

- Vypočítam uhol pod ktorým by čiara zvierala X-ovú os, pretože táto čiara zatiaľ leží vodorovne.
- Potrebujem ešte čiaru otočíť, lenže vlastnosť CSS rotation otáča objekty podľa ich stredu. Preto musím nájsť stred (sx, sy) a vypočítať súradnice x a y ktoré by mala čiara vo vodorovnej polohe (pred otočením).
- Tento blok nakoniec vložím do bloku "relations"na vypočítané súradnice x a y ich pevne vložím cez CSS vlastnosť position a pomocou vlastnosti rotation otočím o vypočítaný uhol.



Obr. 3.16: Vypočítam súradnice x, y a uhol alfa. Potom vložím na súradnice x, y a potom ho otočím o daný uhol. Čierna čiara predstavuje čiaru pred otočením a červená už výslednú čiaru čo sme potrebovali dostať.

Kapitola 4

Implementácia

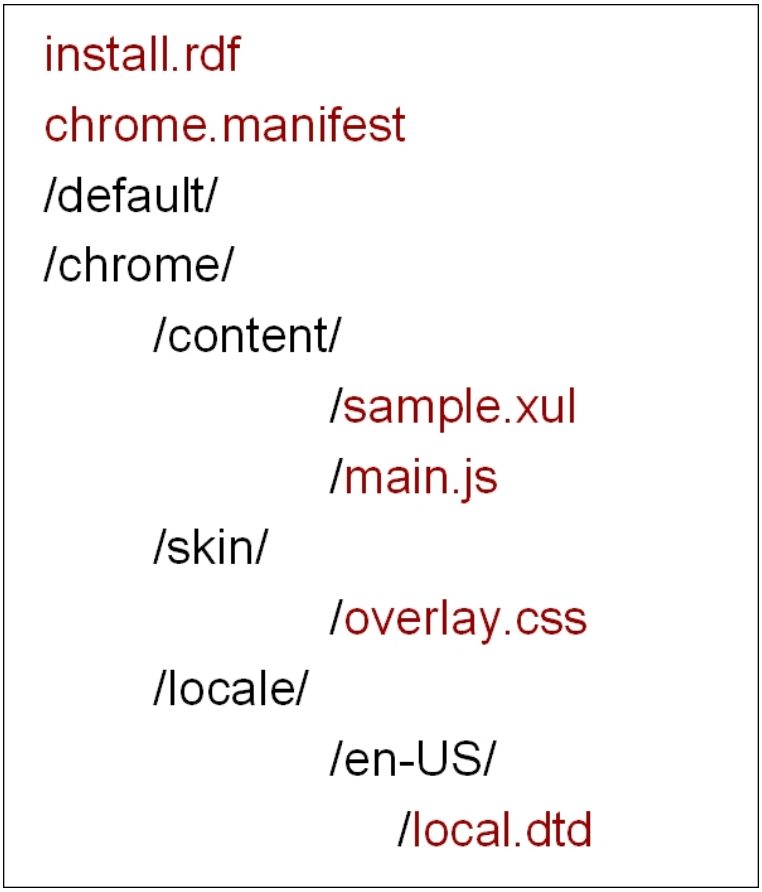
4.1 Tvorba Doplnkov

Pre lepšie pochopenie implementácie Wikinavigátoru, napíšem v tejto kapitole niečo o tvorbe takýchto doplnkov.

4.1.1 Štruktúra Doplnku

Doplnky do Firefoxu sú štruktúry priečinkou a súborov rôzneho typu, zbalené do inštalačného súboru. V tejto hierarchii bežia sú umiestnené javascripty, XUL súbory a iné. Takáto štruktúra priečinkov a súborov môže vyzerať ako na obrázku 4.1. Podrobnejšie popíšem jednotlivé súbory a priečinky :

- Súbor **install.rdf** je súbor v ktorom pomocou určenej štruktúry XML popisujeme meno doplnku, popis, autorove meno, verziu, maximálnu a minimálnu verziu Firefoxu pre spustenie doplnku, domovskú stránku a ďalšie veci.
- V súbore **chrome.manifest** definujem pre hlavné scripty, css súubory a XUL súbory pomocou určenej syntaxe absolútnu cestu, pod ktorou k nim môžem v doplnku pristupovať. Táto cesta pre súbor "chrome/content/sample.xul"bude "chrome://myExtension/content/sample.xul". Ďalej v tomto súbore definujem overlays, ktoré popíšem v samostatnej kapitole.
- V priečinku **default** sú uložené takzvané referencie. V nich si developer môže ukladať nejaké krátke používateľove nastavenia napr. ako home URL a podobne.



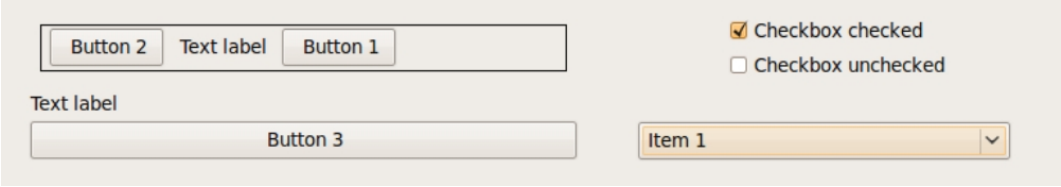
Obr. 4.1: Príklad štruktúry priečinkov a súborov v doplnku. Čierne sú priečinky a hnedé súbry.

- Priečinku **chrome** je hlavný priečinok z ktorého bežia všetky skripty, sú tam zadefinované všetky overlays a súbory z neho môžu mať už spomínanú absolútnu adresu.
- V priečinku **content** sú uložené všetky javascripty, html, XUL alebo jar súubory. Tu sa definuje všetko, čo používateľ vidí pod doplnkom.
- V **skin** sú CSS súbory ktoré definujú vlastnosti objektov.
- V **locale** sú uložene .dtd súubory, ktoré popisújú textové polia v rôznych jazykoch.

V rámci jedného doplnku funguje pristupovanie k súborom aj pomocou relatívnej cesty. Pomocou absolútnej môžu napríklad k súborom pristupovať používaťelia priamo cez Firefox (na mieste kde sa zadáva URL).

4.1.2 XUL

XUL je značkovací jazyk, ktorý vytvorila a používa Mozilla Foundation. Tento jazyk je založený na základe XML štruktúry a popisuje rôzne elementy vo Firefoxe (obrázok 4.3). Sú to všetky buttony, boxy, menu, statusbary, textboxy, atď. (obrázok 4.2).



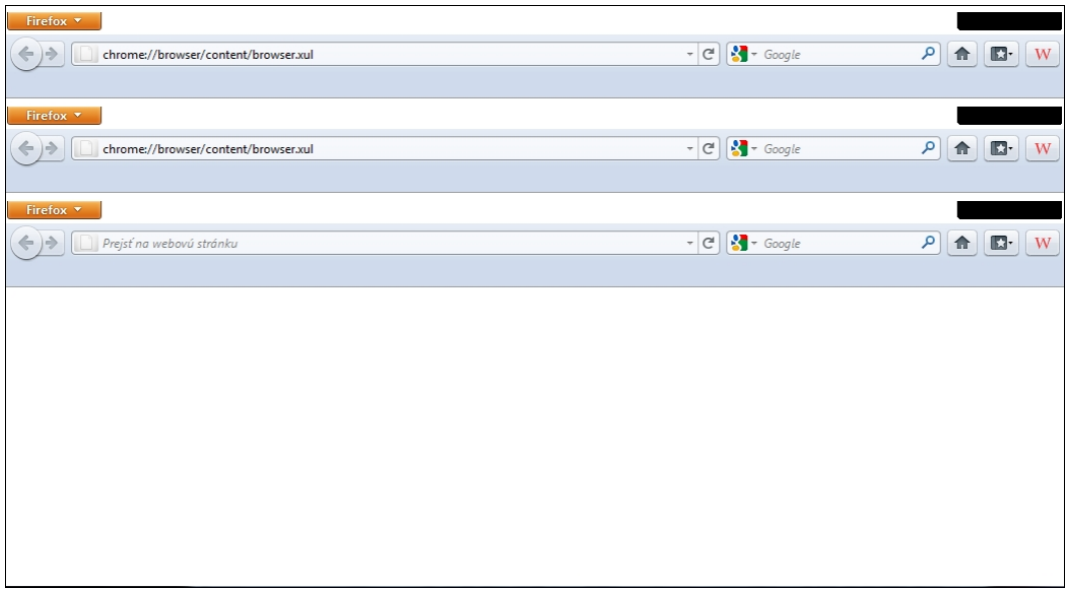
Obr. 4.2: Niektoré XUL elementy vo Firefoxe. (*https://developer.mozilla.org/En/Firefox_addons_developer_guide/Introduction_to_XUL—How_to_build_a_more_intuitive_UI* (25.5.2011))

```
<toolbar>
  <toolbarbutton label="Checkbox Type" type="checkbox" image="firefox.png"/>
  <toolbarbutton label="Menu Type" type="menu" popup="button-popup" image="firefox.png"/>
  <toolbarbutton label="Menu Button Type" type="menu-button" popup="button-popup" image="firefox.png"/>
  <menupopup id="button-popup">
    <menuitem label="Item 1"/>
    <menuitem label="Item 2"/>
    <menuitem label="Item 3"/>
  </menupopup>
</toolbar>
```

Obr. 4.3: Ukážka XUL syntaxe. (*https://developer.mozilla.org/En/Firefox_addons_developer_guide/Introduction_to_XUL—How_to_build_a_more_intuitive_UI* (25.5.2011))

4.1.3 Overlays

Aby sa moje XUL elementy, JavaScripty, CSS a iné súbory zobrazili vo Firefoxe, musím ich niekde zadefinovať. Dá sa povedať že ich pripojím k súborom ktoré zbehnú vo Firefoxe. Hovorí sa tomu, že vytvorím **overlay**. Všetky overlays sú definované v súbore chrome.manifest podľa určenej syntaxe. XUL overlay (napr.: "chrome://myExtension/content/overlay.xul") pripojím za súbor "chrome://browser/content/browser.xul"čo je XUL súbor v ktorom sú zadefinované všetky XUL elementy Firefoxu a bežia v ňom všetky scripty. Pre ukážku že to tak naozaj funguje, stačí ako adresu do prehliadača zadať túto relatívnu adresu (obrázok 4.4). Ďalej ešte môžeme definovať CSS overlay, pomocou ktorej sa dá meniť vzhľad celého Firefoxu. Pre JavaScripty doplnku, ktoré majú bežať pod Firefoxom neexistuje overlay, ale vkladajú sa do XUL súboru, ktorý už overlay je.



Obr. 4.4: browser.xul - Dva krát po sebe som zadal do Firefoxu adresu "chrome://browser/content/browser.xul".

4.1.4 JavaScript Namespacing

Programátori vo svojich doplnkoch používajú rôzne názvy pre svoje funkcie, premenné v javascripte, pre id XUL elementov. Keďže kód doplnku je pripojený rovno za kód Firefoxu (a aj za kódy iných doplnkov), je potrebné dávať unikátne mená. Ľahko by sa totiž mohlo stať že by nejaká funkcia nepracovala správne, kryla by sa s firefoxovou funkciou alebo s inými doplnkami. Unikátne mená je najlepšie riešiť nejakým unikátnym prefixom, čo je však dosť nepraktické a nepohodlné pre programátora. Preto sa asponň v JavaScriptovom kóde oplatí používať tzv. **Namespacing**, čo zaručí všetkým funkciám a premenným unikátne meno.

4.1.5 Inštalácia

Na inštaláciu doplnku používa Firefox **XPI** technológiu, ktorú vyvýja a používa Mozilla Foundation. Vyslovuje sa to 'zippy' a je to aj kôli tomu, pretože inštalačný súbor vzniká nasledovne:

- Všetky súbory v opísanej hierarchii zbalím do súboru zip
- Príponu .zip prepíšem na .xpi

Teraz už len stačí tento súbor uploadnuť niekde na web. Keď zadáme jeho adresu do Firefoxu, spýta sa či chcem nainštalovať tento doplnok. Nainštaluje ho a po reštarte prehliadača by už mal normálne fungovať. Ak si vo Firefoxe naklikáme informácie o doplnku, zobrazia sa niektoré veci ktoré sme definovali v súbore install.rdf.

4.2 Implementácia Wikinavigátoru

4.2.1 Štruktúra

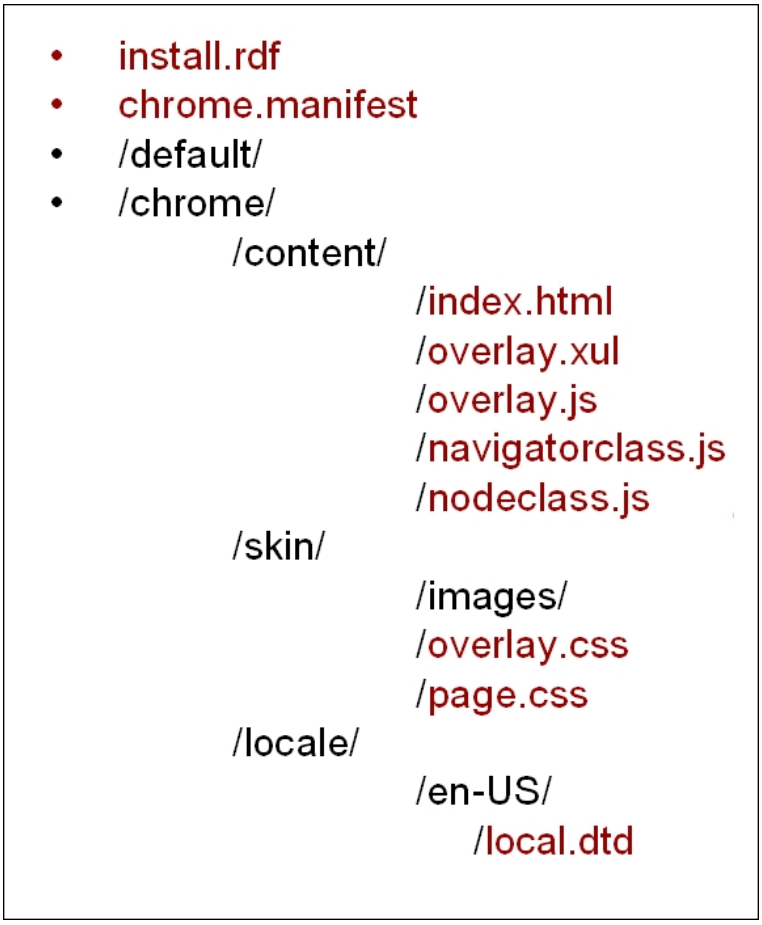
V kapitole štruktúra doplnkov som popísal ako by mala vyzerať názorná štruktúra. Štruktúra Wikinavigátoru je znázornená na obrázku 4.5. Teraz popíšem čo robia a na čo slúžia súbory v mojom doplnku :

- Súbor **install.rdf** obsahuje informácie o doplnku ako bolo popísané skôr.

- V **chrome.manifest** sú okrem definícií absolútnych ciest, definované aj dve hlavné overlay - **overlay.xul** a **overlay.css**
- **overlay.xul** je hlavný súbor v ktorom je popísaný button ktorý som do Wiki-navigátoru pridal a je umiestnený na dané miesto. Ďalej je tu nalinkovaný hlavný script **overlay.js** a ostatné scripty
- Súbor **overlay.js** je hlavný script, v ktorom spúšťam všetky funkcie čo bežia pod browserom.
- **navigatorclass.js** popisuje triedu navigator.
- **nodeclass.js** popisuje triedu node.
- Súbor **index.html** je hlavná HTML stránka doplnku, v ktorom sa vykresľuje celý strom a ostatné veci. Je v ňom nalinkovaný súbor **page.css**.
- **page.css** definuje kaskádové štýly hlavnej vykresľovacej stránky.
- **overlay.css** definuje kaskádové štýly XUL elementu (v mojom prípade tlačidla s logom).
- V **locale.dtd** sú popísaná jazyková varianta.

4.2.2 Vytvorenie tlačidla vo Firefoxe

Mojou prvou úlohou bolo vytvoriť tlačidlo a umiestniť ho niekde do Firefoxu. Podľa určitej syntaxe v XUL som definoval nový button v súbore **overlay.xul**. Atribútom ïnsertbefore"sa potom vloží pred nejaký element ktorý už vo Firefoxe je. Na zistenie aké majú elementy ID, aké majú vlastnosti a kam by som moje tlačidlo mohol vložiť som použil **DOM Inspector** (obrázok 4.6). Je to doplnok do Firefoxu, ktorý rozkladá rôzne súbory (napr. xml, html, xul, atď) na základe ich XML štruktúry. Tieto sú pekne vyznačené v náhľade stránky, dajá sa zistiť aké majú atribúty a hodnoty. Zobrazil som preto súbor **browser.xul**, tam som si našiel vhodný element pred ktorý by som ho vložil. Konečné umiestnenie je vidno na obrázku 4.7.

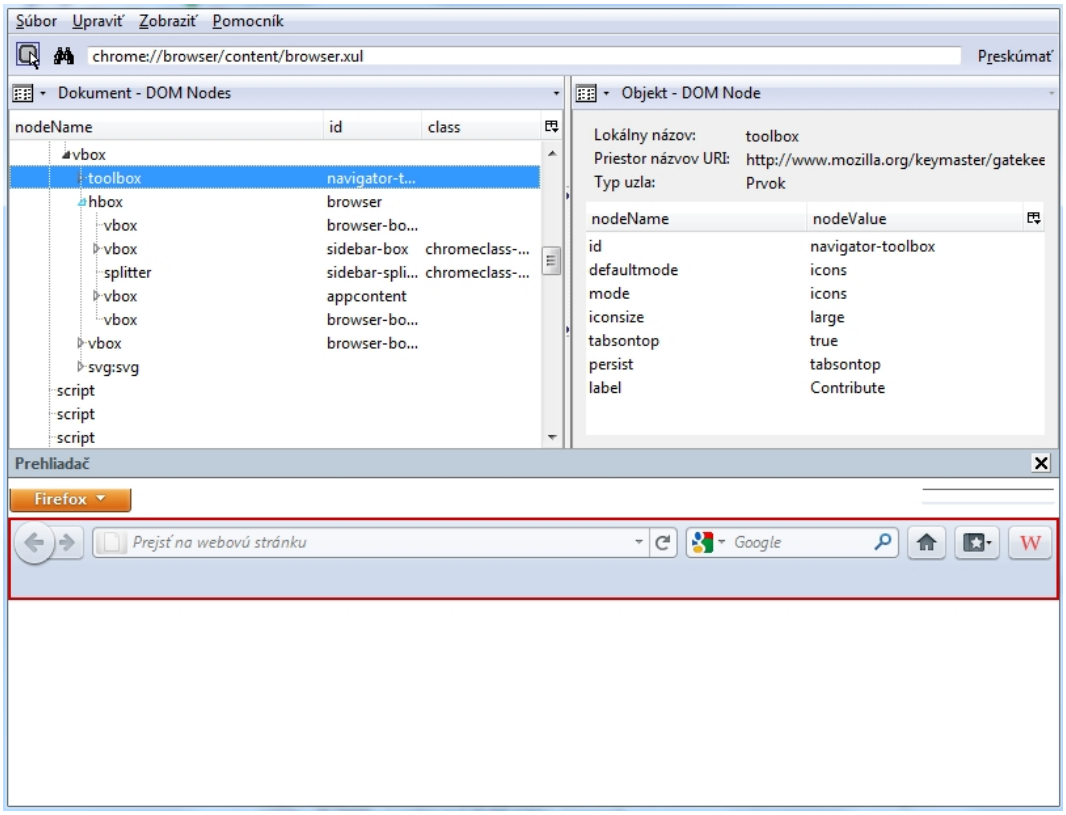


Obr. 4.5: Štruktúra Wikinavigátoru

4.2.3 Beh programu

Hlavný JavaScript **overlay.js** je spustený pod prehliadačom a beží po celý čas. Všetky objekty, funkcie a premenné z ktorými robím, sú pod namespace "Wikinavigator". Hlavné funkcie, ktoré zaisťujú beh Wikinavigátoru sú v sekcii Namespacu "Wikinavigator.Main". Triedy Navigator a Node sú pod "Wikinavigator.Classes". Teraz popíšem ako môj script beží :

- Najprv som dal listery na otvorenie nového tabu. Ak sa tam načíta Wikinavigátor, tak odstránim celý toolbar aj s riadkom pre zadávanie adries a načíta sa **index.html**.
- Ak sa načíta iná stránka, tak sa skontroluje, či už nieje ako aktívna bunka v poli



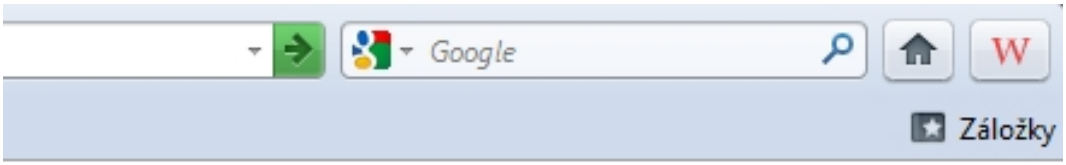
Obr. 4.6: DOM Inspector

zatvorených tabov. Ak je, tak sa len obnoví objekt Navigator príslušný tomuto tabu. Ak nieje tak sa vytvorí nový objekt Navigator, aj s nejakým ID, ktorému prislúcha nejaký browser. Tento objekt pridám do poľa aktívnych tabov.

- Pre lepšie porozumenie cez objekt **gBrowser**, ktorý je vo Firefoxe, pristupujem takmer k celému prehliadaču. Môžem zistit "tab container"čo je nejaké pole tabov. Každý tab má svoj objekt **browser** a objekt **window**, každé so svojimi vlastnosťami. Cez browser tabu môžem pristupovať priamo k stránke načítanej v tomto tabe (cez browser.contentDocument, alebo browser.contentWindow).

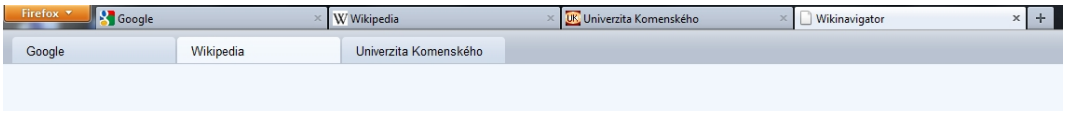
- Ďalej sa pridá listener na zatvorenie tabu. Vtedy sa z poľa aktívnych tabov presunie príslušný objekt Navigator do poľa zavretých tabov.

- Ďalšia vec ktorú som spravil pri otvorení nového tabu je, že dám listner na PageLoad v tomto browseri.



Obr. 4.7: Tlačidlo Wikinavigátoru - v pravo hore W

- Ak sa loadne stránka v tabe, tak sa skontroluje či už nieje v deťoch aktívnej bunky, nieje parent, alebo nieje tá istá. Ak niečo z toho platí, tak sa len zmení aktávna bunka a ak nie, tak sa pridá nové dieťa aktívnej bunke a zmení sa na aktívnu.
- Každej novej bunke sa prida OnClick listener, na presmerovanie do príslušného tabu, s príslušnou URL
- Pri zobrazení stránky Wikinavigátoru, chrome://Wikinavigator/content/index.html, pomocou DOM aktualizujem hornú navigáciu aktívnych a zavretých tabov (obr.4.8) a tiež vykreslený strom daného tabu.
- Prekresľovanie sa volá vždy keď dám focus na stránku Wikinavigátoru, keď sa nejaká stránka loadne a focus je na stránke Wikinavigátoru a keď mením zobrazovaný tab v hornej navigácii tejto stránky.

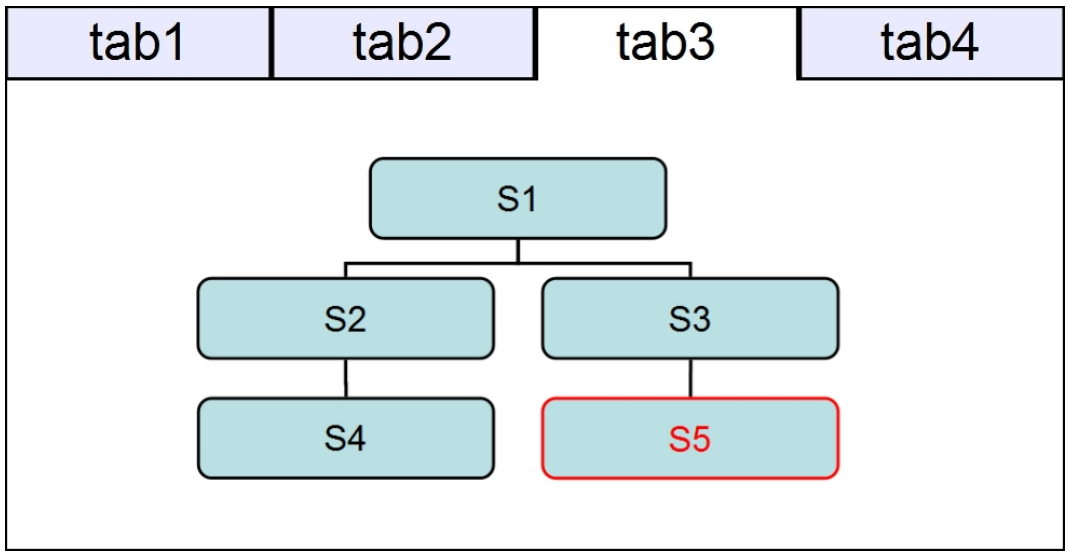


Obr. 4.8: Navigácia v rámci Wikinavigátoru

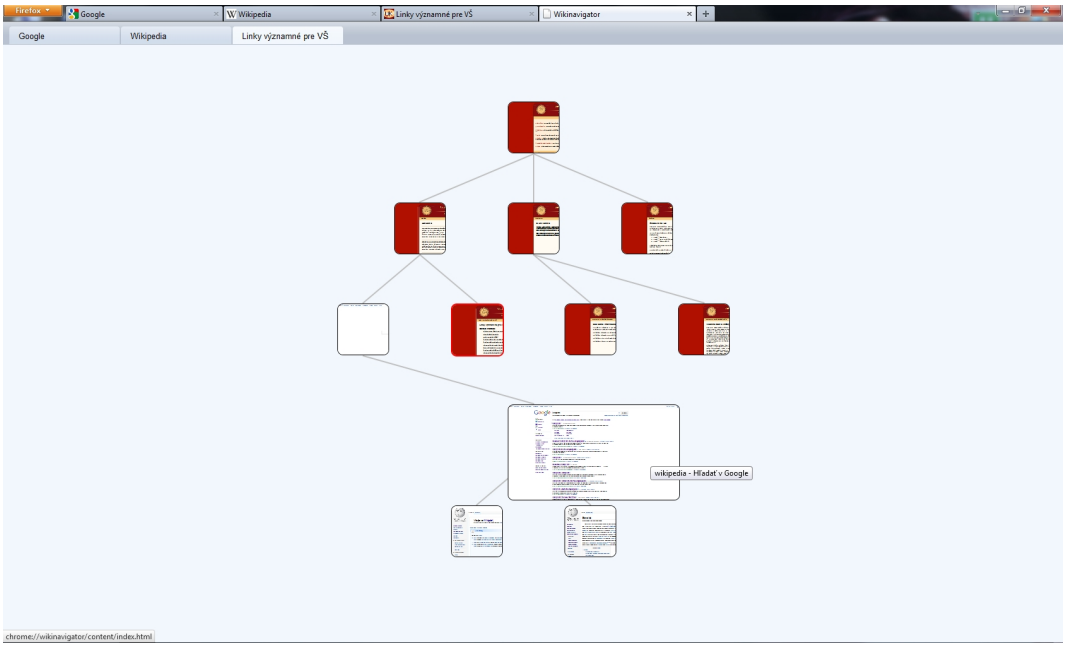
4.3 Otázka implementácie histórie bunky

Každá bunka má nejakú históriu, podľa ktorej sme sa k nej prepracovali. Túto históriu reprezentujú všetky adresy buniek, ktoré sú na najkratšej ceste z našej bunky až do koreňa stromu. Pri prekliku na bunku vo Wikinavigátore sa však história stromu

líši od histórie tabu po ktorej môžeme chodiť pomocou tlačidiel History Back a History Forward (obrázok 4.9). História prekliknutej bunky sa dá implementovať aj do tabu vo Firefoxe, ale otázka je, či nebude mať potom používateľ v tomto zmätok. Zatiaľ som to neimplementoval, ale časom možno pridám túto možnosť do nastavení Wikinavigátoru.



Obr. 4.9: Z aktívnej stránky S5 sa po stlačšní tlačidla back dostanem na stránky S3 a potom S1 (budem to označovať že má History(S3, S1)). Ak by som klikol na bunku S4 (stane sa aktívnou), potom bude mať History(S5, S3, S1). Nápad na implementovanie histórie bunky znamená, že po tomto prekliku by mala aktívna bunka S4 History(S2, S1).



Obr. 4.10: Konečná podoba Wikinavigátoru.

Záver

Doplnok Wikinavigátor som už dokončil vo svojej prvej verzii. Je naprogramovaný pre Firefox 4.0, ale funguje od verzie 3.6 a vyššie. Obsahuje základnú funkcionalitu a to je vykresľovanie histórie jednotlivých tabov. Okrem toho zobrazuje históriu aj neaktívnych (zatvorených tabov).

Tento doplnok som testoval a používal na svojom počítači. Zjednoduchšil mi browsovanie a veľa krát mi pomohol dostrať sa na stránky, ktoré som navštívil a stratil zo základnej back a forward histórie (mŕtve bunky o ktorých som písal). Priebežne budem tento doplnok vyvíjať, implementovať ďalšie nápady na vylepšenie, napríklad:

- Pridať ukladanie histórie. Wikinavigátor v terajšej verzii totiž po reštarťe Firefoxu zabudne všetky dáta a začne s odpočúvaním tabov nanovo.
- Možnosť sledovať len domény(to znamená sledovať len adresy typu*www.nieco.com* a nie všetky podadresy napr. *www.nieco.com/podstranka.html*). Pre niektorých ľudí nieje až tak dôležité kde presne boli, ale akú doménu navštívili. V budúcnosti by Wikinavigátor mohol mať v nastaveniach checkbox, ktorým by si používateľ vybral aký mód prehliadania chce.
- Implementovať mazanie buniek a podstromov.
- Možnosť zvolenia módu s implementáciou už spomínanej histórie bunky.

Dúfam že Wikinavigátor bude užítočný nielen mne, ale aj iným používateľom Firefoxu, ktorí si ho budú môcť stiahnuť.

Literatúra

[20001] Pat Hanrahan. 2001. Tree drawing algorithms. In Berkley Business College [online], 2001. Dostupné na internete: http://graphics.stanford.edu/courses/cs448b-02-winter/lectures/treesgraphs/tree.graph.pdf.

[20006] Diaz Dustin. 2006. Namespacing your javascript. In dustindiaz.com [online], 2006. Dostupné na internete: http://www.dustindiaz.com/namespace-your-javascript/.

[20009] Nyman Robert. 2009. How to develop a firefox extension. In Add-ons Blog [online], 2009. Dostupné na internete: http://blog.mozilla.com/addons/2009/01/28/how-to-develop-a-firefox-extension/.

[20110a] Adrian Rusu. 2010. Trees and graphs. Handbook of Graph Drawing and Visualization [online], 2010. Dostupné na internete: http://graphics.stanford.edu/courses/cs448b-02-winter/lectures/treesgraphs/tree.graph.pdf. ISBN: 1584884126.

[20110b] Mozilla Developer Network Editors. 2010. Firefox addons developer guide. In Mozilla Developer Network [online], 2010. Dostupné na internete: https://developer.mozilla.org/En/Firefox_addons_developer_guide.

[20111a] Mozilla Developer Network Editors. 2011. Building an extension. In Mozilla Developer Network [online], 2011. Dostupné na internete: https://developer.mozilla.org/en/Building_an_Extension.

[20111b] Mozilla Developer Network Editors. 2011. Tabbed browser. In Mozilla Developer Network [online], 2011. Dostupné na internete: https://developer.mozilla.org/en/Code_snippets/Tabbed_browser.

[Gro05a] LCC 3401 Firefox Group. Firefox extension development tutorial :: Configuration filesl. 2005. In Rietta [online], 2005. Dostupné na internete: http://www.rietta.com/firefox/Tutorial/conf.pdf.

[Gro05b] LCC 3401 Firefox Group. Firefox extension development tutorial. 2005. In Rietta [online], 2005. Dostupné na internete: http://www.rietta.com/firefox/Tutorial/backend.pdf/.