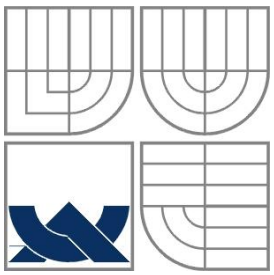
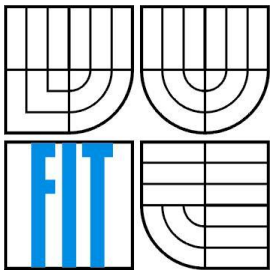


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

PODPORA A VYUŽITÍ MODERNÍCH WEBOVÝCH STANDARDŮ

SUPPORT AND UTILISATION OF MODERN WEB STANDARDS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Lukáš Švihel
AUTHOR

VEDOUCÍ PRÁCE — Ing. Svatopluk Šperka
SUPERVISOR

BRNO 2011

Zadání bakalářské práce/13224/2010/xsvihe02

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2010/2011

Zadání bakalářské práce

Řešitel: **Švihel Lukáš**
Obor: Informační technologie
Téma: **Podpora a využití moderních webových standardů**
Support and Utilisation of Modern Web Standards
Kategorie: Web

Pokyny:

1. Nastudujte současné návrhy standardů HTML5 a CSS3 - především z hlediska toho, o co rozšiřují své předchůdce.
2. Zmapujte současnou podporu standardů v aktuálních verzích webový prohlížečů (vemte v úvahu i testovací verze).
3. Na základě získaných znalostí navrhněte netriviální komponentu (či komponenty), která bude demonstrovat a prověří možnosti standardů z hlediska realizace komplexních vizuálních a interaktivních vlastností.
4. Navrženou komponentu (komponenty) implementujte.
5. Poznatky zdokumentujte a nastiňte možná řešení či nemožnost řešení některých problémů v rámci starších standardů.
6. Zhodnoťte možnosti, které nové standardy přinášejí.

Literatura:

- http://dev.w3.org/html5/spec/Overview.html
- http://www.w3.org/TR/css3-roadmap/
- http://www.w3.org/Style/CSS/Specs/All

Při obhajobě semestrální části projektu je požadováno:

- Bez požadavků.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Šperka Svatopluk, Ing.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2010
Datum odevzdání: 27. července 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

V práci se věnuji problematice využití moderních webových technologií. Vývoji historie, srovnání s předchozí verzí. Popisuji všechny důležité aspekty prvků, jež HTML5 a CSS3 přináší a vysvětluji jejich přínos a účel. Možnosti nových webových standardů demonstruje webová aplikace. Tato aplikace je webovým prohlížečem obrázků z databáze flickr.com s možností editace, uložení, offline režimu a efekty zpříjemňující uživatelské rozhraní.

Abstract

This bachelor's thesis describes use of modern web standards. History of development compared with old version. Also describe every important aspects of elements in HTML5 and CSS3 with acquisition and purpose. The result of this thesis is example web application build on HTML5 and CSS3 which demonstrate their potentiality. This application is web browser of pictures from flickr.com with choise of edit and saves, offline mode and effect improve rich user experience.

Klíčová slova

HTML5, CSS3, Web Storage, Local Storage, Session Storage, Offline aplikace, W3C, WHATWG, Multimédia, Video, Audio, Canvas, SVG, Web sockets, manifest, prvek

Keywords

HTML5, CSS3, Web Storage, Local Storage, Session Storage, Offline application, W3C, WHATWG, Multimedia, Video, Audio, Canvas, SVG, Web sockets, manifest, element, tag

Citace

Lukáš Švihel: Podpora a využití moderních webových standardů, bakalářská práce, Brno, FIT VUT v Brně, 2011

Podpora a využití moderních webových standardů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Svatopluka Šperky.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Lukáš Švihel
27.7.2011

Poděkování

V této sekci je možno uvést poděkování vedoucímu práce ing. Svatopluku Šperkovi za jeho odbornou pomoc a užitečné rady, které mi během řešení práce poskytl.

Obsah

1 Úvod

Webové technologie od svého zrodu v roce 1990 prošly velkými změnami a za poslední desetiletí se masově rozšířily do celého světa. Společně s rostoucími možnostmi přístupu k internetu a rychlosti připojení rostou i požadavky uživatelů po náročnějších a dynamičtějších webových aplikacích. Webová stránka již dnes mnohdy není statickou prezentací, ale dynamickou oboustranně komunikující webovou aplikací. A v tomto trendu pokračují i nově nastupující moderní webové standardy, jež funkcionalitu webových stránek značně rozšiřují a tvorbu příjemného uživatelského prostředí zjednodušují.

Tato práce se zabývá podporou a možnostmi moderních webových standardů v době psaní práce a to konkrétně specifikacemi HTML5 a CSS3. Tyto specifikace jsou prozatím ve fázích rozpracování a tudíž nejsou hotovy. Již dnes ale obsahují celou řadu prvků, které lze využít. Čtenář se nejprve seznámí s historií a vývojem webových technologií společně s uvedením důležitých jmen a skupin, které hrály důležitou roli. Dozví se o historii vzniku W3C konsorcia, které spolupracuje s veřejností na vývoji webových standardů.

V kapitole 3 jsou porovnávány tyto nové standardy HTML5 a CSS3 se svými předchůdci - stávajícími standardy. Nejprve přímým porovnáním a vysvětlení důvodů u prvků měnících se oproti staršímu standardu. Poté uvedením nových prvků, které HTML5 přináší. I kaskádové styly byly porovnány s předchůdcem rozborem vlastností, které jsou ve stádiu alespoň jisté malé podpory.

Praktickým výsledkem této práce je webová aplikace, která demonstruje možnosti a použitelnost nových webových standardů a to propojením těch nejdospělejších částí specifikace HTML5 a CSS3. Webová aplikace – prohlížeč obrázků zobrazuje náhodné obrázky z veřejné webové databáze *Flickr.com*, zobrazuje je v uživatelsky přívětivém prostředí, umožňuje základní editaci uložení do lokální paměti a následné prohlížení upravených uložených obrázků i v režimu *offline*. Kapitola 4 se zabývá vlastní implementaci této aplikace a probírá úskalí, které nastaly při její tvorbě. V této kapitole jsou také porovnány teoretická s reálnou podporou prohlížečů společně s možností či nemožností řešení problémů v rámci starších standardů.

2 Historie webových standardů

Tato kapitola se věnuje historii a vývoji webových technologií. Čtenář se dozví o vzniku první webové stránky společně s bližšími informace a důvody vzniku organizací jako je známé W3C konsorcium a ECMA zabývající se webovými standardy.

2.1 Cesta vývoje webových technologií

V roce 1990 vznikla první webová stránka a to jako součást projektu WWW vznikajícího v CERNu[1]. Tento projekt se zabýval problémy se sdílením informací ve velkých institucích, jakou právě CERN byl. Na projektu pracoval Brit Tim Berners-Lee a své výsledky dvouletého projektu popsal v dokumentu „*HTML Tags"*, kde také uvedl několik elementů, které by mohly být použity při psaní webových stránek. Zajímavostí bylo i to, že nebyla předpokládána znalost jazyka (verze 0.9) a tedy byl společně s prvním webovým prohlížečem vyvinut i první editor webových stránek.

V roce 1994 byla vytvořena první standardizovaná verze HTML2.0, která zachovávala prvky z předchozí verze, ale také se rozšiřovala o prvky nové jako formuláře a podporu grafiky. Také jako první verze HTML odpovídá syntaxi SGML. Ve stejném roce odchází Tim Berners-Lee z organizace CERN a zakládá sdružení W3C[2]. První snahou oddělit vzhled dokumentu od jeho struktury a obsahu bylo CSS1 v roce 1996. Bohužel tehdejším majoritním prohlížeče, jako *Internet Explorer 3* a *Netscape Navigator*, styly CSS příliš kvalitně nepodporovaly.

JavaScript byl standardizován asociací ECMA[3] roku 1997 a organizací ISO[4] v roce 1998.

Začátku roku 1997 byla vydána verze HTML3.2, jež je přepracovaným návrhem verze 3.0, který se nikdy nestal standardem z důvodu přílišné složitosti. Verze 3.2 byla také první verzí standardizovanou výhradně W3C konsorciem. Na konci téhož roku, byla zveřejněna verze 4.0. Od té doby se stav značkovacího jazyka HTML příliš nezměnil.

Druhá verze kaskádových stylů CSS2 přináší v roce 1998 několik nových možností, jako například „polohování" prvků pomocí `absolute`, `relative` a `z-index`. Konečná verze byla ustanovena až v roce 2000. CSS2 se v dnešní době nepoužívá jen k formátování HTML ale například i k formátování XML dokumentů.

V roce 2005 přichází technologie AJAX[5]. Jedná se o technologii asynchronně komunikujícího *JavaScriptu* a XML. AJAX propojuje jazyk HTML (či XHTML[6]) s CSS, *JavaScript*

[1] Evropská mezinárodní organizace pro jaderný se sídlem v Ženevě
[2] World Wide Web Consortium mezinárodní konsorcium vyvíjející webové standardy
[3] European Computer Manufacturers Association
[4] International Organization for Standardization
[5] Asynchronous JavaScript and XML
[6] Extensible HyperText Markup Language

a *XMLHttpRequest* pro asynchronní komunikaci mezi klientem a serverem. Příchodem roku 2008 byl zveřejněn návrh HTML5 od W3C.

2.2 Konsorcium W3C a HTML5

Jak bylo zmíněno výše, toto sdružení již existuje od roku 1994 a zakladatelem, dnešním ředitelem je Tim Berners-Lee. Jedná se o mezinárodní konsorcium, jehož hlavní úkolem je dohlížet na vývoj standardů na internetu. W3C není jedna velká společnost. Skládá se z několika institucí ať už akademického nebo komerčního původu. V době založení odehrálo konsorcium důležitou úlohu, a to sjednocení velkého množství různých nestandardních postupů, jež prohlížeče používaly. Toto způsobovalo velké problémy při programování webových stránek při snaze zachovávat stejný vhled na všech prohlížečích. Tlak na výrobce prohlížečů, aby dodržovali W3C doporučení, se ještě zvýšil v roce 1999 vydáním HTML4.0 a s tím spojenou stabilizací syntaxe a struktury HTML.

Každý vydaný standard (neboli doporučení) musí projít několika stádii vývoje [1] než je ve finále „doporučen“. A to postupně od pracovního návrhu „*Working Draft“*, přes „*Candidate Recommendation“* a „*Proposed Recommendation“* až po konečné vydání W3C konsorciem. Společně s doporučeními vydává taký řadu poznámek *notes*, které jsou pouze informačního charakteru.

Na začátku roku 2011 také bylo zveřejněno HTML5 oficiální logo (Obrázek 2.1), které je k dispozici ke stažení a použití pro každého, kdo chce zdůraznit podporu tohoto standardu na svém webu. Logo je k dispozici pod licencí Crative Commons 3.0[7].

Obrázek 2.1 - Oficiální logo HTML5 [23]

Začátek HTML5 můžeme přiřadit k roku 2004, kdy firma Adobe uspořádala seminář zabývající se webovými aplikacemi a složenými dokumenty, se zaměřením na budoucnost webových aplikací. W3C ovšem ve stejnou dobu pracovalo na jiném směru vývoje webových standardů. Rozhodlo zaměřit spíše na vývoj XHTML. Ovšem několik vývojářů z firem Mozzila, Opera a Apple vyvíjející webové prohlížeče, požádala o zahrnutí několika zmodernizovaných prvků, pro podporu webových aplikací, do standardu HTML 4. Jejich návrh byl ale zamítnut. Tato skupina se, ovšem nenechala neúspěchem odradit a vytvořila oficiální skupinu známou dnes pod zkratkou WHATWG neboli „*Web Hypertext Application Technology Working Group*“ a ve vývoji pokračovala dál konkrétně na tom co je nám dnes známo jako HTML5. Již v roce 2005 publikovali svůj první pracovní návrh pod jménem „*Web Applications 1.0*“. V roce 2006 konsorcium W3C ustoupilo od dalšího vývoje XHTML a domluvilo

[7] Dílo smíte šířit - kopírovat, distribuovat a sdělovat dílo veřejnosti a také upravovat - pozměňovat, doplňovat, využívat celé nebo částečně v jiných dílech [20].

se na spolupráci s WHATWG. Název specifikace byl následně přejmenován na HTML5. Od té doby W3C i WHATWG pravidelně do specifikace přispívají.

V roce 2008, Ian Hickson editor HTML5 specifikace, uvedl v rozhovoru, že předpokládá dosažení „*Candidate Recommendation*“ statusu roku 2012, finálního testování v roce 2020 a konečné „*Proposed Recommendation*“ [1] verze doporučení v roce 2022. Je ovšem rozdíl mezi finálním doporučení W3C a skutečným možným využití v praxi na majoritních prohlížečích. HTML5 postupovalo ve vývoji postupně po celou dobu, ale koncem roku 2009 až 2010 společnosti Google a Apple vývoj urychlily. Co se týká podpory na mobilních platformách i velkým zvětšením zájmu veřejnosti. HTML5 je v době psaní této práce v pracovní fázi „*Last Call*“. Některé funkce jsou více vyspělé než jiné. Je již řada prvků, které mohou vývojáři a tvůrci webu využít již dnes. Navíc podpora prohlížečů roste každou novou vydanou verzí.

3 Srovnání HTML5 a CSS3 s předchozími verzemi

Jak již z předchozí kapitoly 2 vyplývá tak HTML5 je prozatím stále vyvíjejícím se standardem a bylo by velmi vhodné jej porovnat a srovnat se standardem předchozím. Čtenář se dozví jaké změny a nové možnosti HTML5 oproti předchůdci HTML4 přináší. Také i kaskádové styly CSS3 budou porovnány s CSS2.

3.1 Porovnání HTML5 a HTML4

3.1.1 Základní prvky

Základními prvky HTML, které tvoří základ dokumentu jsou `<!DOCTYPE>` určující typ dokumentu, `<html>` kořenový prvek a prvky `<head>`, `<meta>`, `<style>` a `<script>`, které definují vlastnosti dokumentu. Nová specifikace vnáší do těchto prvků jistá zjednodušení. Asi nejlepším porozuměním co HTML5 je, bude cestou porovnání se svým předchůdcem, předchozí verzí HTML4.

Rozdílu při psaní HTML5 dokumentu si všimneme hned na prvním řádku, kde musí být umístěn `<!DOCTYPE>`. Jedná se o pozůstatek z minulosti s možností přepínat prohlížeč do různých kompatibilních zobrazovacích režimů. Zde máme HTML4 `<!DOCTYPE>` a to jeden z mnoha možných:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

Naproti tomu HTML5 má velmi krátký zápis. Delší není potřebný, jelikož HTML5 není založeno na SGML[8]. Příklad HTML5 `DOCTYPE`:

```
<!DOCTYPE HTML>
```

HTML strukturovaný kód dokumentu je reprezentován stromem. Strom se rozvětvuje od kořenu na větve a tyto větve se mohou dále znovu rozvětvit. Na konci větve nalezneme list – samotný prvek. Tomuto kdy uzly stromu odpovídají jednotlivým prvkům dokumentu se říká DOM[9]. Díky stromové

[8] Standard Generalized Markup Language - je standardní jazyk určený k formálnímu popisu struktury dokumentů.
[9] Z anglického Document Object Model – objektový model dokumentu.

struktuře dovedeme rozlišit nadřazené (rodičovské) prvky, podřazené (děti) prvky. Kořenovým prvek HTML dokumentu je `<html>`. Tento prvek se díky HTML5 mění a zjednoduší. Takto vypadá zápis HTML4 kořenového prvku:

```
<html xmlns=http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
```

Tolik vlastností není dle specifikace HTML5 potřeba. V případě `xml:lang` by se jednalo o duplicitní definování jazyka (jazyk HTML a XML stejný) a tedy ho nemusíme uvádět. A v případě druhém se u `xmlns` jedná o nadbytečné definování, jelikož prvky HTML5 jsou vždy v XHTML ve jmenném prostoru[10]. Zápis v HTML5 je tedy opět poněkud kratším:

```
<html lang="en">
```

Zjednodušením prošel i prvek `<meta>` definující informace o dokumentu jako klíčová slova, popis dokumentu, autora apod. V HTML4 se znaková sada dokumentu zapisuje následovně:

```
<meta http-equiv="Content-Type" content="text/html"; charset="UTF-8" />
```

Oproti kratšímu zápisu v HTML5 pokud se chceme spolehnout na zadání znakové sady z HTTP hlavičky[11]:

```
<meta charset="UTF-8" />
```

Zjednodušení se v HTML5 také dočkalo vložení CSS stylu, za pomocí prvku `<style>`. Oproti delšímu zápisu v HTML4 není třeba u HTML5 uvádět `type`. Ten je totiž ve výchozím stavu nastaven jako `„text/css"`. Rozdíl vypadá takto:

```
<style type="text/css"></style>                    <!—HTML4-->
<style></style>                                    <!—HTML5-->
```

Zkrácení zápisu je aplikováno i na prvek `<script>` umožňující vkládání skriptů (většinou JavaScript). Vlastnost `type` je ve výchozím stavu nastavena na `text/javascript`. Zkrácení vypadá takto:

```
<script type="text/javascript"></script>           <!—HTML4-->
<script></script>                                  <!—HTML5-->
```

[10] Jmenný prostor odpovídá použitému programovacímu jazyku (neboli prostor prvků).
[11] HTTP hlavička je součástí (popis) každého HTTP požadavku či odpovědi.

3.1.2 Význam rozšiřující prvky

Již HTML4 poskytovala prvky, které pomáhaly vnést do obsahu webu strukturu. Ale téměř žádné z nich nedodávaly význam. Dokument byl sice rozčleněn, ale jednotlivé části nemusely nést o sobě jednoznačnou a nezaměnitelnou informace. V praxi je hojně využíváno prvků `<div>` a `<span>`, které pomocí identifikátorů nebo tříd mohou být přistupovány z Javascriptu nebo stylovány CSS vlastnostmi. Toto řešení ale nepřidává žádnou sémantiku stránky a tedy nevíme například zda-li prvek `<div id="info">` obsahuje informace o stránce nebo o něčem jiném.

K tomu, aby mohl mít autor větší kontrolu nad sémantikou stránky, přichází na pomoc standard HTML5 s řadou nových prvků. Na mnoha místech bude možné nahradit prvky `<div>` za mnohem smysluplnější bloky s předem daným významem. Toto usnadní nejen čitelnost stránky vyhledávačům, webovým prohlížečům na mobilních zařízeních, ale také i nástrojům usnadňující přístup pro nevidomé, neslyšící atd.

Začněme s novým prvkem `<section>` u kterého specifikace HTML5 říká: „*Prvek `<section>` představuje sekci dokumentu, typicky s názvem nebo hlavičkou.*“ [2]. Sekcemi mohou býti např. kapitoly, popisky či jiné informace. Tento prvek má podle popisu velmi podobnou funkčnost jako prvek `<div>`. Ovšem na rozdíl od něj odděluje významový obsah dokumentu visuálního oddělení u prvku `<div>`. Příklad použití:

```
<section>
	<h1>Informace o osobě</h1>
	<ul>
	<li>Jméno a Přijmení</li>
		<li>Telefon</li>
		<li>...</li>
	</ul>
</section>
```

Mezi podobný prvek patří i `<article>`. Specifikace tento prvek popisuje následovně: *„Prvek `<article>` představuje část obsahu, která tvoří nezávislou část dokumentu nebo stránky. Například časopis nebo novinový článek nebo příspěvek na blogu.“* [3]. Důležitou informací je slovní spojení *„nezávislou část“*. A tedy je vhodné jej použít u významově oddělených bloků. Prvek `<aside>` reprezentuje sekci na stránce, jejíž obsah je naopak související s okolím tohoto prvku. Umístění `<aside>` je důležité, protože by mělo být vnořeno v obsahu, s kterým souvisí. Pokud bude umístěn nevnořen, bude souviset s celým dokumentem. Tento prvek je výborný pro použití např. u citací nebo také u seznamu podobných článků. Každá sekce by měla mít i svůj nadpis. Pokud budeme chtít použít vícero nadpisů `<h1>` až `<h6>` popisující různě důležité informace (např. jména autora, datum publikování apod.), je vhodné použít prvek `<hgroup>`, jež nám tuto skupinu uzavře.

Pro část na konci dokumentu nazývající se patičkou máme také nový prvek jména `<footer>`, který typicky obsahuje informace jako příbuzné dokumenty, copyright apod. Obvykle se tento prvek nachází se na konci sekce, s kterou je spřažen a tedy nemusí plnit funkci jen patičky stránky, ale taktéž např. funkci patičky článku nebo soupisu informací. Prvek `<nav>` podle anglického „*navigation*“ nám významově oddělí tu část, jež odkazuje na jiné části stránky nebo na jiné další stránky. Při použití tohoto prvku je důležité uvědomit si, že slouží jen k hlavní navigaci na stránce.

V případě, kdy bychom chtěli vložit obsah (obrázek, graf, ukázkový příklad), jež jeho odstranění neovlivní chápání významu, ale přesto jeho obsah souvisí s hlavním obsahem dokumentu, je vhodné použit prvek `<figure>` společně s `<figcaption>`, který obsahuje jeho titulek, jako například:

```
<figure>
	<figcaption>Naše společné zážitky</figcaption>
	<img src="norsko_v_lete.jpg" width="640" height="480" />
</figure>
```

Při vypisování detailů okolního obsahu, které mohou být v případě potřeby schovány či naopak zobrazeny, je dle nové specifikace HTML5 nejvhodnější využít prvek `<details>`. Specifikace také doporučuje jeho použití s prvkem `<summary>`, který obsahuje nadpis. Jako například:

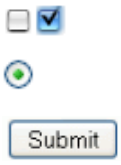
```
<details>
	<summary>Copyright 2000.</summary>
	<p>Vlastníkem stránek je FIRMA7 s.r.o.</p>
</details>
```

Společným využitím prvku `<command>`, jež je v HTML5 přidán nově a prvku `<menu>`, který byl zastaralý a specifikace HTML5 jej definuje nově, můžeme docílit několika typů seznamů prvků. Jako celek lze tuto množinu prvků použit jako panel nástrojů nebo obsahová nabídka. Nastavení vlastnosti `type` prvku `<menu>` může nabývat následujících hodnot:

`list` ..... Výchozí. Specifikuje menu jako list.
`context` ..... Specifikuje kontextové menu.
`toolbar` ..... Specifikuje menu jako panel nástrojů.

Uvnitř je nejvhodnější využít prvku `<command>`, specifického tlačítka. Pokud tento prvek umístíme mimo prvky`<menu>`, tak nebude zobrazen. Toto tlačítko je ovládacím tlačítko podle typu vlastnosti `type`.

Ukázka zobrazení typů tlačítek `command` ve webovém prohlížeči Google Chrome verze 12:




```
<command type="radio">
<command type="checkbox">
<command type="command">
```

Dále můžeme nastavit vlastnost `checked` a nadefinovat tím aktuální hodnotu zaškrtnutí (výchozí hodnota nezaškrtnuto - `false`). Vlastností `disabled` můžeme tlačítko zakázat a znemožnit funkci. Důležitou vlastností je `icon`, která umožňuje definovat adresu obrázku, jež se zobrazí jako tlačítko. Ukázkové použití:

```
<menu>
      <command type="checkbox" checked="checked">Text</command>
      <command type="checkbox">Tady</command>
</menu>
```

Takto tedy vypadají nové prvky zvyšující význam jednotlivých částí HTML dokumentu. Porovnání prvků HTML4 a prvků zvýrazňující význam z HTML5 se nachází na Obrázek 3.1 - Zápis struktury HTML4 a HTML5. Jedná se o vizuální zobrazení prvků HTML dokumentu s hlavičkou obsahující hlavní nadpis společně s vedlejším. To vše ve skupině nadpisů a prvku `<header>`. Dále navigace `<nav>` a sekce `<section>` obsahující jednotlivé články `<article>` s titulkem využívajíc opět `<header>` a sekcí s obsahem. Na konci se nachází postranní panel `<aside>` a `<footer>` s copyrightem.

Obrázek 3.1 - Zápis struktury HTML4 a HTML5

Mezi užitečný prvek pro označení textu např. po vyhledávání slouží `<mark>`. A opět se tedy snižuje potřeba prvků `<div>` a zvyšuje význam jednotlivých částí dokumentu. Např:

```
<p>Neuvěřitelně tajné <mark>slovo</mark>, které nikdy nikdo nenajde.</p>
```

Mezi prvek neurčující význam ale zlepšující čitelnost patří prvek `<wbr>`. Tento prvek umožňuje „rozlomit“ velmi dlouhé slovo nebo dlouhý řetězec bez mezer pro lepší čitelnost. V mnoha případech např. na zařízeních s malou obrazovkou se takto velmi dlouhá slova dostanou např. mimo obrazovku. Tento prvek určuje v kterém místě se může takto dlouhé slovo zalomit na nový řádek v případě jeli toto potřeba. Jako na příkladu:

```
<p>Hipopoto<wbr>monstro<wbr>seskvi<wbr>pedaliofobie je fóbie z dlouhých slov.</p>
```

Prvek `<time>` pomáhá definovat datum a čas ve tvaru dobře čitelném pro člověka. Vlastnost `datetime` specifikuje datum nebo čas pro tento prvek.
Například:

```
<time datetime="2009-09-22">Dnes</time>
```

V případech zobrazení znaků/písma východoasijské kultury je doporučeno využít prvku `<ruby>` společně s `<rt>` a `<rp>` vysvětlující význam nebo výslovnost znaků a prvek.

3.1.3 Formulářové prvky

Řadu let se vývojáři soustředili na způsob, jak vytvořit více interaktivní prostředí, které zvýší komfort používání. K tomuto začali využívat CSS styly a technologie jako JavaScript nebo Adobe Flash či Microsoft Silverlight[12]. Specifikace HTML5 přichází s řadou vylepšení, která mají tyto nedostatky zmírnit. Je důležité také zmínit, že zobrazení nových vstupních i výstupních prvků záleží na interpretaci prohlížeče a můžeme se setkat s mírnými odlišnostmi jako u prvku na Obrázek 3.2 nebo Obrázek 3.3 - Zobrazení `<input type="date">` v prohlížeči Opera verze 11.10..

Pro zobrazení průběhu ve známém rozsahu a dílčích hodnotách slouží prvek `<meter>`. Ukázkové použití:

```
<meter value="3" min="0" max="10">Otázka 3 z 10</meter>
```

Parametr `value` představuje zobrazovanou hodnotu. Parametrem `min` a `max` nastavíme rozsah dílčích částí a možných stavů. Hodnota vlastnosti `value` nemůžu nabývat vyššího čísla než maximální hodnota určená parametrem `max`. Pokud nezadáme parametry `min` a `max` tak se můžeme v hodnotou `value` pohybovat v rozmezí `0-1`, což jako v příkladu výše může znamenat procentuální část. Ukázkové použití:

```
<meter value="0.3">30%</meter>
```

Mezi poslední doplňující parametry patří `low`, `high` a `optimum`. Parametr `low` reprezentuje bod v zobrazeném měření, který je považován za nízký a patřičně zvýrazněný. Parametr `high` za vysoký a

[12] Silverlight je platforma společnosti Microsoft, která je určena pro vývoj multimediálních aplikací.

parametr `optimum` za optimální. Tyto tři zmíněné parametry se používají společně. Níže uvedený příklad ukáže slovy obtížněji vysvětlitelný význam nejlépe. Prohlížeče tyto hodnoty vizuálně rozdílně zobrazí:

```
<meter value="90" min="0" max="100" low="40" high="90" optimum="100">
```

Zobrazení  při hodnotě `value="10"`.

Zobrazení při hodnotě `value="50"`.

Zobrazení  při hodnotě `value="90"`.

Prvek `<progress>` je podobný jako předchozí `<meter>` ovšem dle specifikace reprezentuje stav úkonu, kde není známa přesná koncová hodnota. Ukázkové použití:

```
<progress>76 položek načteno</progress>
```

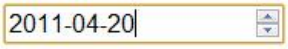
Mezi možné parametry patří volitelný parametr `max`, který definuje hodnotu dokončení a parametr `value`, jež je aktuální hodnota.

Základním vstupním prvkem kde je uživateli umožněno zadávat data formuláře je `<input>`. Základní struktura formuláře, dle specifikace HTML4, může vypadat např. takto:

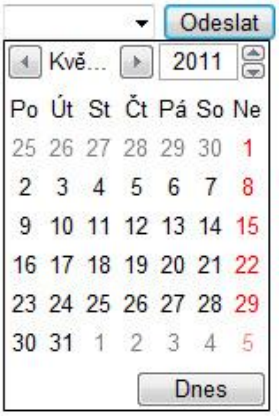
```
<form id="mujFormular" action="#"method="post" >
      <label for="url">Adresa</label>
      <input type="text" id="url" tabindex="10">

      <label for="přijmení">Email</label>
      <input type="text" id="email" tabindex="20">

      <label for="text">Poznamka:</label>
      <textarea type="textarea" id="text" tabindex="30"></textarea>

      <input type="submit" id="submit" tabindex="40" value="Odešli">
</form>
```

Takto vytvořený formulář `<form>` nám obsahuje dva prvky `<input>` s vlastností `type` jako `text`. Toto jsou standardní formulářové buňky pro zadání textu (první pro zadání webové adresy a druhá pro zadání emailové adresy). Dále pak prvek `<textarea>` představující rámeček pro zadávání mnohořádkového textu. Každý vstupní prvek pro zadávání dat má před sebou prvek `<label>`, jež obsahuje zobrazitelný text popisující uživateli typ dat, které má vepsat. Na konci formuláře se nachází tlačítko pro odeslání, jež je také prvek `<input>`, ale s parametrem `type` jako `submit`. Dále každý prvek má zadanou vlastnost `tabindex`, která specifikuje pořadí prvků v případě navigace za pomocí klávesy **tab**. Standard HTML5 přináší v oblasti formulářů nemalá vylepšení. Pokud bychom chtěli v našem případě striktně určit, které prvky jsou povinné a musí být vyplněny před odesláním

formuláře, použijeme novou vlastnost `required`. A to například takto: `<input type="text" required>`. Další možností je využití nových typů prvku `<input>`. Při zadání vlastnosti `type="email"` tak nejen, že formulář přijme jen validní email, ale také mohou jistá zařízení např. mobilní zařízení při zadávání vstupu reagovat a např. nabídnout uživateli vhodnější rozložení kláves na jeho dotykovém přístroji. Pro zlepšení uživatelského prostředí zadáme vlastnost `type="date"` kde uživatel bude moci vybrat přesné datum z výběru.



Obrázek 3.2 – Zobrazení `<input type="date">` v prohlížeči Google Chrome verze 12.



Obrázek 3.3 - Zobrazení `<input type="date">` v prohlížeči Opera verze 11.10.

Při potřebě zadání dat pomocí posuvníku, je možnost využití vlastnosti `type="range"`. Po zadání minimální vlastnosti `min` a maximální `max` bude prvek `<input>` zobrazen jako posuvník.

Obrázek 3.4 - Zobrazení `<input type="range">` v prohlížeči Google Chrome verze 12.

Vlastnost `type="search"` je určena pro zadání výrazu vyhledávání. Vlastnost `type="tel"` definuje možná vstupní data jako telefonní číslo. Pokud vlastnost `type` bude zadána jako `type="color"` bude tento prvek `<input>` zobrazen jako výběr barvy. Veškeré tyto nové formulářové prvky a vlastnosti také zdůrazňují sémantiku prvků ve formuláři.

3.2 Nové prvky v HTML5

Standard HTML5 přináší i nové prvky a s tím spojené nové funkce, které doposud nebylo možné využít. Jako náhrady se využívalo doplňků třetích stran (např. Flash) pro specifické pokročilé funkce popsané dále v kapitole.

3.2.1 Multimediální prvky

Ve specifikaci HTML5 se nachází řada nových multimediálních prvků, které umožňující vložit například multimediální data bez potřeby zásuvných modulů nebo přídavných programů v prohlížeči. Technika vkládání multimediálních objektů je doposud nejvíc známá u vkládání obrázků prvkem `<img>` na webové stránky/aplikace. Ve starším standardu HTML4 bylo vkládání audio/video dat bez použití doplňku pod názvem Flash[13] velmi komplikované. HTML5 toto razantně zjednodušuje.

Doposud v žádné z HTML specifikací nebyl přímo prvek pro přehrávání video obsahu na stránce. Prvek `<video>`, který přináší specifikace HTML5 umožňuje vkládat video mezi obsah společně s jeho kontrolními prvky. Příklad použití:

```
<video src="film.ogg" controls="controls"></video>
```

Obrázek 3.5 – Vzhled vloženého videa v prohlížeči Google Chrome verze 12.

Hlavními vlastnostmi tohoto prvku je zdroj videa `src` jež může mít 3 různé formáty (první sloupec tabulky Tabulka 3.1 – Podporované formáty HTML5 videa v majoritních prohlížečích [21]. Tento seznam podpory je seznamem aktuální podpory v době psaní práce – přesněji 16.7.2011. Prvním je `Ogg` jež je prozatím podporován v prohlížečích Firefox, Opera a Chrome. Mezi další patří `MPEG4` využívající `H.264` video kodek. Tento formát podporuje prohlížeč IE9[14] a Chrome společně se Safari. Posledním třetím kodekem je `WebM` jež je podporován prohlížeči Firefox, Chrome a Opera.

[13] Doplněk firmy Adobe, který umožňuje vložení interaktivních animací, prezentací na webu.
[14] Internet Explorer 9

Formáty videa	IE	Firefox	Opera	Chrome	Safari
Theora+Vorbis+Ogg	-	3.5+	10.5+	5.0+	-
H.264+AAC+MP4	9.0+	-	-	5.0+	3.0+
WebM	-	4.0+	10.6+	6.0+	-

Tabulka 3.1 – Podporované formáty HTML5 videa v majoritních prohlížečích [21].

Prvek disponuje řadou vlastností pro individuální nastavení. Vlastnost `audio` může být nastavena na hodnotu ztlumeno „`muted`" a umožňuje video ztlumit. Vlastnost `autoplay` nadefinuje video tak, že je automaticky spuštěno ihned po načtení. Vlastnost `controls` vyvolá zobrazení ovládacího panelu ve videu (Obrázek 3.5 – Vzhled vloženého videa v prohlížeči Google Chrome verze 12.). Vlastnost `loop` představuje nastavení videa do režimu nekonečného přehrávání. Vlastnost `poster` specifikuje `url` obrázku reprezentující video před jeho spuštěním. Poslední vlastnost `preload` umožňuje nenačíst video po načtení stránky.

Prvek `<audio>` reprezentuje zvukový proud, který lze na webové stránce pomocí ovládacích prvků přehrávat, posouvat či nastavit hlasitost (

Obrázek 3.6 – Výchozí zobrazení prvku `<audio>` v prohlížeči Google Chrome). Příklad použití:

```
<audio src="music.ogg" controls="controls">
      Váš prohlížeč si nerozumí s prvkem audio.
</audio>
```

Obrázek 3.6 – Výchozí zobrazení prvku `<audio>` v prohlížeči Google Chrome verze 12.

Takto vytvořený prvek obsahuje vlastnost `src` obsahující cestu ke zdroji. Dále volitelná vlastnost `controls` slouží k zobrazení ovládacích prvků (podobně jako u prvku `<video>`). Obsah nacházející se uvnitř prvku bude zobrazen v prohlížečích, které tento prvek nepodporují. Mezi další vlastnosti patří `loop` pro nekonečné opakování a `preload`, jež umožňuje audio nenačíst po načtení stránky.

Jelikož u obou dvou prvků `<audio>` i `<video>` je možné vkládat data v různých formátech a každý prohlížeč podporuje jinou množinu formátů, tak toto může znamenat velký problém z hlediska kompatibility stránky. Pokud budeme chtít videa přehrát kterémukoliv uživateli, tak bude nejvhodnější využít nový prvek v HTML5, a to `<source>` pomocí něhož nyní můžeme vkládat video

a audio ve více formátech a prohlížeč použije ten, který podporuje. Takto vypadá způsob zápisu:

```
<video controls="controls">
      <source src="matrix.ogg" type="audio/ogg" />
      <source src="matrix.mp3" type="audio/mpeg" />
      Váš prohlížeč si v tímto videem nerozumí!
</video>
```

Pokud bychom chtěli vložit jiný multimediální obsah, můžeme toto provést pomocí prvku `<embed>`. Obecně je tento prvek používán pro integraci externí aplikace nebo jiného interaktivního obsahu (obvykle nepsaném v HTML). Přestože je již ve velké míře používán (např. vkládání Adobe Flash) již dlouho dobu, tak tento prvek nenalezneme v žádné ze předchozích specifikací HTML. Příklad použití:

```
<embed src="flash.swf" />
```

3.2.2 Grafické prvky

Novou funkcí v HTML5 je i grafická reprezentace kreslícího plátna `<canvas>`. Tento prvek reprezentuje bitmapové plátno, závislé na rozlišení, které může být použito jako dynamické vykreslování obrazu. Plátno neslouží jen k zobrazení, ale jeho hlavním důvodem je upravování této oblasti dle potřeby. Vytvoření prvku `<canvas>` může vypadat následovně:

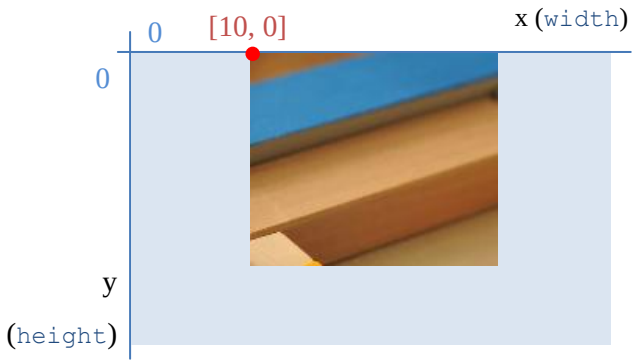
```
<canvas id="myCanvas" width="800px" height="600">
      Tento prohlížeč nepodporuje canvas.
</canvas>
```

Takto vytvořený prvek má definován `id` pro identifikaci při kreslení do plátna. Další zadané vlastnosti jsou šířka `width` a výška `height`. Obsah uvnitř uzavřeného prvku `<canvas>` bude zobrazen, jen pokud prohlížeč nepodporuje tento prvek. Pro kreslení na plátno použijeme funkce `canvas` API[15] a JavaScript kód. Takto může vypadat jednoduché vykreslení obrázku na plátno:

[15] Zkratka anglického Application Programming Interface jež označuje rozhraní pro tvorbu aplikací.

```
<script>
	function kresli () {
		var canvas = document.getElementById('myCanvas');
		var contex = canvas.getContext('2d');
		var obraz = new Image();
		obraz.src = 'img/obraz.jpg';
		obraz.onload = function() {
			contex.drawImage(obraz, 10, 0);
		}
	}
</script>
```

Tato vytvořená funkce `kresli()` kreslící obrázek do plátna, nejdříve do proměnné `canvas` uloží odkaz na dříve vytvořené plátno. Plátno získáme funkcí `getElementById` a `id` identifikátorem v prvku `canvas`. Poté vytvoří tzv. kontext, do něhož probíhá samotné kreslení. V tomto případě kreslíme do 2D kontextu. Lze kreslit i 3D tělesa do 3D kontextu, která jsou vykreslována WebGL[16]. Vytvořením nového objektu `Image` reprezentující obrázek a přiřazením cesty `src` ke zdroji je obrázek připraven. Pomocí události `onload`, která se vyvolá při načtení, provedeme vykreslení obrázku metodou `drawImage(img, x, y, width, height)`. Tato metoda umístí obrázek na souřadnice `x` a `y`. Volitelně je možné zadat i `width` šířku a `height` výšku obrázku. Počátek soustavy souřadnic `canvas` je v levém horním rohu podle Obrázek 3.7 – Zobrazení soustavy souřadnic plátna `canvas`.



Obrázek 3.7 – Zobrazení soustavy souřadnic plátna `canvas`.

Je možno použít mnoho způsobů kreslení na plátno. Např. metoda `beginPath()` inicializuje kreslení cesty (případně zruší cestu starou). Metoda `fillRect(xBegin, Ybegin, width, height)` vytvoří cestu čtyřúhelníkového tvaru ze zadaného bodu a zadané šířce a výšce. Metoda `fillStyle(color)` změní barvu cesty. Metoda `moveTo(x, y)` slouží k posunutí výchozího bodu kreslení o zadané `x` a `y`.

[16] Grafická knihovna pro web.

Metoda `lineTo(x, y)` slouží ke kreslení čáry z výchozího bodu do bodu relativně zadaného bodu pomocí `x` a `y`. A také metoda `closePath()` pro uzavření cesty.

Porovnejme `<canvas>` společně s `<svg>` na Obrázek 3.8. SVG je zkratka anglického Scalable Vector Graphics, česky škálovatelná vektorová grafika a umožňuje vekreslení obrazu vytvořeného z tvarů a křivek. Oproti tomu u prvku `<canvas>` se jedná o rastrovou grafiku (pixelově orientovanou), a tudíž veškeré operace probíhají nad sadou pixelů. Z tohoto plyne několik výhod a nevýhod pro `<canvas>`. Vykreslovací výkon je stále stejný bez ohledu na počet vložených prvků na plátno, jelikož počet zobrazovaných pixelů je stále stejný. Výsledné plátno lze uložit. Vhodný pro přímý přístup k jednotlivým pixelům. Jako nevýhody lze uvést: není zde možné využít jakýchkoliv integrovaných animací. Nevhodné při použití, kdy je potřeba reagovat na podněty uživatele.

Obrázek 3.8 - Ukázka porovnání bitmapové grafiky s vektorovou. Obrázek převzat z [22].

3.2.3 Prvky offline režimu

Obecně si pod zkratkou API můžeme představit soubor procedur, funkcí či tříd nějaké knihovny, které můžeme využívat k programování vlastních komplexních aplikací. V HTML5 je nově několik rozhraní umožňující nám přístup k novým funkcím.

Jednou z nových možností jak vylepšit vaší webovou aplikaci a dát ji nové možnosti je použít HTML5 uživatelskou paměť pro uložení důležitých souborů pro následnou možnost pracovat i v režimu *offline*. Režim *offline* je režim, kdy prohlížeč nemá k dispozici připojení k internetu, a tudíž ani k serveru, kde jsou webové stránky uloženy a požadavky od klienta vyřizovány.

Jako první podmínkou pro základní použití je vytvoření `manifest` souboru, který bude obsahovat list souborů, jež mají být uloženy do uživatelské paměti prohlížeče. Např. soubor `cache.manifest`:

```
CACHE MANIFEST
```

```
CACHE:
img/cat.jpg
img/car.jpg
script.js
demo.css
```

Toto je list souborů, jež budou načteny v *online* a budou použity, jakmile budeme *offline*. Povinný řádkem je `CACHE MANIFEST` dále potom soubor může obsahovat soubory pod hlavičkou `CACHE`. Tyto soubory, jež jsou v explicitní sekci a měly by být přidány do cache a upřednostněny před *online* obsahem. Dva soubory na jednom řádku pod hlavičkou `FALLBACK` budou považovány za náhradu (druhý za první) při režimu offline.

```
FALLBACK:
img/online_status.jpg img/offline_status.jpg
```

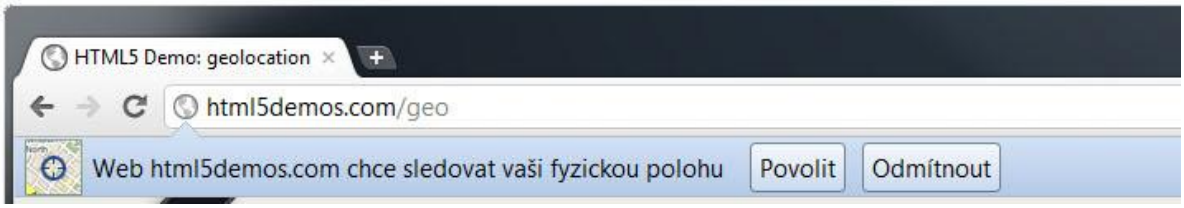
Soubory pod hlavičkou NETWORK patří do seznamu výjimek a nebudou načteny. Nyní je nutno přiřadit cestu k `manifest` souboru k prvku `<html>` a to například následovně:

```
<html manifest="cache.manifest">
```

Pro korektní rozpoznání souboru `manifest` musí i server tento soubor rozpoznat a odeslat vhodnou odpověď (HTTP Request) s hlavičkou obsahující `Content-Type: text/cache-manifest`. Pro odchytávání událostí spojené s *offline* web aplikacemi využijeme události v JavaScriptu jako `checking`, `noupdate`, `downloading`, `progress`, `updateready`, `obsolete` nebo `error`. Využití možnosti *offline* webových aplikací může být užitečným přínosem v případě, kdy je potřeba přistupovat k datům, informacím apod. i v době kdy není k dispozici připojení k internetu.

3.2.4 Prvky geolokačního API

Geolokační API není přímo součástí specifikace HTML5, ale patří mezi moderní webové technologie a je mnohdy s HTML5 nesprávně spojováno. Toto API umožňuje webovým aplikacím poskytnout polohu uživatele. Toto je mnohem přesnější způsob zjišťování polohy, než v minulosti využívaná lokalizace dle IP adresy. Uživatel je před poskytnutí dat o jeho poloze informován a vyzván k potvrzení nebo zamítnutí (oObrázek 3.9 – Požadavek na získání polohy v prohlížeče Google Chrome 12.).



Obrázek 3.9 – Požadavek na získání polohy v prohlížeče Google Chrome 12.

Jádro geolokačního API je postaveno na globálním objektu `navigator.geolocation`. Pro jednorázové získání polohy slouží metody tohoto objektu `getCurrentPosition()`. Oproti tomu metoda `watchPosition()` vytvoří tzv. watcher (hlídač) umožňující aplikacím získávání polohy za

určitou časovou periodu. Příklad jednoduché funkce pro získání dat:

```
if(navigator.geolocation) {
      navigator.geolocation.getCurrentPosition(plotLocation, error);
} else {
      // geolokace není dostupná
}
```

Nejprve zjistíme, jestli objekt geolocation existuje. Pokud neexistuje, geolokace není prohlížečem podporována. Pokud existuje, získáme data. Funkce vepsány jako parametry v metodě `getCurrentPosition` složí k jednoduchému odlišení chování při úspěšném získání dat (funkce `plotLocation`) či neúspěšném získání dat (funkce `error`). Při získání polohy bude vrácená funkce obsahovat dva parametry, `coords` obsahující veškeré data polohy, rychlosti pohybu, nadmořské výšky, přesnost apod. a `timestamp` obsahuje časové razítko těchto dat. Použití těchto dat může být základem pro webové služby postavené na aktuální poloze uživatele. Hlavním potenciálem jsou také mobilní zařízení, u kterých se může stát poloha velmi užitečnou.

3.2.5 Prvky uživatelské paměti

Pokud webová aplikace vyžaduje ukládání data na uživatelské straně, tak aktuální možnosti jsou poněkud limitující. Využitím *cookies*, která umožňují uložení malého množství dat (maximálně 4kB), můžeme právě narazit na problém s nedostatečným prostorem pro data, ale také s vlastností *cookies* a to, že jsou tyto data posílána v každém HTTP požadavku. Hlavním využitím *Web Storage* je v možnosti uložit velké množství dat na straně klienta. Původně specifikace popisující *Web Storage* byla součástí HTML5 specifikace, ale nyní je již osamocena ve své vlastní specifikaci [4]. Data jsou uložena v párech (klíč – hodnota). *Web Storage* je navrženo pro dvě odlišné skupiny dat:

- *Session Storage* – Umožňuje ukládání dat a kontrolu nad nimi v jediné `session`[17] a samotném okně.

- *Local Storage* – Kontroluje data přístupná skrze všechny okna a ta, jež se ukládají na delší použití než je jedna `session`.

Mezi dvě hlavní funkce patří přiřazení dat ke klíči pomocí metody `getItem(key)` pro načtení dat dle zadaného klíče a `setItem(key, data)` a nastavení dat se zadaným klíčem. Příklady použití práce v *JavaScript* funkci s *Local Storage*:

```
var nacteneJmeno = localStorage.getItem("jmeno");
// udělej něco po načtení z Local Storage
localStorage.setItem("jmeno ", Josef);
```

Podobně bude probíhat i práce s *Session Storage*:

```
var nacteneJmeno = sessionStorage.getItem("jmeno");
// udělej něco po načtení z Session Storage
sessionStorage.setItem("jmeno ", Josef);
```

Alternativní zápis přes hranaté závorky je `var nacteneJmeno = localStorage["jmeno"];` Mezi další užitečné metody pro práci s *Web Storage* jsou `removeItem(key)` pro vymazání potřebného klíče, `clear()` pro vymazání všech párů klíč-data a vlastnost `length` vracející počet přiřazených páru dat.

3.2.6 Prvky komunikace se serverem

Další specifikací, která byla původně součástí HTML5 a poté se od ní odtrhla do své samostatné [5] jsou tzv. *Web Sockets*. Jedná se o pokročilou komunikaci mezi klientem (webovým prohlížečem) a serverem. Jako nejzákladnější metoda je zasíláním požadavků serveru, který tento požadavek zpracuje a pošle nazpět. Obvykle toto bez použití HTML5 znamenalo znovunačtení stránky. Tento problém zminimalizovala technologie AJAX, který je schopen posílat data asynchroně[18] pomocí *XMLHttpRequest* objektů a zobrazovat data přijatá od serveru bez nutnosti znovunačtení stránky. Toto umožňuje vytvářet mnohem dynamičtější a interaktivnější obsah a uživatelské prostředí. I přesto má AJAX své slabé stránky a pro jistá použití není naprosto vhodný. Důvodem může být omezení, kdy pouze klient může zahájit přenos dat. Navíc mohou nastat s komunikací přes Proxy Servery a

[17] Session představuje permanentní síťové spojení mezi klientem a serverem, zahrnující výměnu paketů.
[18] Data jsou posílána na pozadí mimo načítání stránky

Firewally. *Web Sockets* API umožňují vytvořit samostatného oboustranného spojení mezi klientem a serverem. *Web Sockets* využívají svůj vlastní protokol umožňující bezproblémové průchody přes Firewally apod. Příklad vytvoření spojení v JavaScriptu:

```
var mySocket = new WebSocket("ws://www.demo.com");
```

Takto vytvořené spojení můžeme využívat za pomocí metod a posílat data nebo monitorovat spojení. Jako alternativu k *WebScokets* pro zpětnou kompatibilitu při nepodpoře prohlížeče, můžeme využít open source Java[19] a JavaScript implementaci *Web Sockets* `jWebSocket` [6].

3.3 Zanikající prvky a vlastnosti

Ve specifikaci HTML5 jsou nejen prvky přidávány, ale také prvky rušeny z důvodů zastaralosti či vytvoření prvku nového univerzálnějšího. Nyní budeme soustředit na ty, které již nebudou v novém standardu doporučovány. Například prvek `<acronym>` byl v HTML4 používán pro označení zkratek. Ve specifikaci HTML5 ho již nenajdeme. Náhradou je prvek `<abbr>`. Také prvek `<applet>` již není doporučeno používat. Ve staré specifikaci byl používán pro vložení externího programu napsaného v Javě. Namísto něj raději použijme prvek `<object>`, který má větší možnosti než `<applet>`. I dalších několik prvků, které byly určeny pro formátování textu, či stránky již nový standard nedoporučuje používat. A naopak doporučuje tímto oddělit obsah od vizuální reprezentace a raději použít stylů CSS. Jedním z řady takovýchto prvků je prvek `<basefont>`, který byl v HTML4 používán pro definování výchozí barvy písma nebo druhu písma dokumentu. Doporučuje se raději využít stylování pomocí CSS. Prvkem `<big>` jsme doposud mohli zvýrazňovat text, ale v nové specifikaci je doporučeno použít vhodnější CSS styl jako: `font-weight: bold.` Prvek `<center>` slouží k zarovnání textu na střed. Na toto je také vhodnější využít CSS stylování. Prvek `<dir>` jako „*directory list*“ slouží k stylování seznamu. Dle nové specifikace je doporučeno použít novější `<ul>` prvek. Prvek `<font>` byl v HTML4 používán pro nastavení druhu písma, barvy a velikost písma určitého, tímto prvek ohraničeného, textu. Prvky `<frame>`, `<frameset>` a `<noframes>` byly v HTML4 používány pro vkládání rámů respektive skupiny rámů rozdělující stránku na více částí, do kterých jsou načteny jiné stránky. Prvek `<noframes>` zobrazoval obsažený text těm prohlížečům, co nepodporovaly tyto rámy. Dle nové specifikace nejsou podporovány a to z důvodů špatného dopadu na použití webových stránek. I prvky jako `<s>` neboli `<strike>`, `<tt>`, `<u>` nebo `<xmp>` mají vhodnější náhradu. Přehled v tTabulka 3.2 – Nedoporučované zastaralé prvky a jejich novější interpretace.

[19] Java je objektově orientovaný programovací jazyk, který vyvinula firma Sun Microsystems.

nepodporovaný	náhrada	nepodporovaný	náhrada
<acronym>	<abbr>	<noframes>	-
<applet>	<object>	<tt>	CSS styly
<dir>		<u>	CSS styly
<s>, <strike>	<del>		CSS styly
<xmp>	<pre>	<big>	CSS styly
<frame>	-	<basefont>	CSS styly
<frameset>	-		

Tabulka 3.2 – Nedoporučované zastaralé prvky a jejich novější interpretace.

Specifikace HTML5 přináší také několik změn v odebrání vlastností z některých prvků. Seznam vlastností, které se v HTML5 oproti HTML4 nepoužijí, najdete v Tabulka 3.3 – Nepoužívané vlastnosti prvků z HTML4 v HTML5.

prvek	nepoužívané vlastnosti
a, link	rev, charset
img	longdesc, name
html	version
body	background
table	bgcolor, border, cell padding, cell spacing
tr	bgcolor
th	abbr, height, width
td	scope, bgcolor, height, width

Tabulka 3.3 – Nepoužívané vlastnosti prvků z HTML4 v HTML5.

3.4 Porovnání CSS3 s CSS2

V současnosti je aktuální verzí kaskádových stylů verze 2 přesněji 2.1 zkráceně CSS2. Níže bude stručně popsána čtenáři funkčnost kaskádových stylů a způsob zápisu s příklady u těch případů, které jsou alespoň v malém měřítku podporovány prohlížeči. Následovat bude porovnání co nového vyvíjející se standard CSS3 přináší. Pro bližší porozumění veškerých možností CSS2 doporučuji „*CSS kompletní průvodce*“ [7] společně s webovým průvodcem „Jak psát web“[20].

[20] http://www.jakpsatweb.cz/css/

3.4.1 CSS2

Zkratka CSS znamená zkratku anglického „*Cascading Style Sheets*“. Do češtiny přeloženo jako Kaskádové styly. Kaskádové styly způsobily určitou revoluci v tvorbě webových dokumentů a aplikací a to z důvodu, že umožnily efektivně oddělit obsah od jeho prezentace. Konkrétně tak, že HTML představuje strukturu a obsah webu, a CSS představuje jeho prezentační formu neboli způsob zobrazení. Oddělení stylů CSS od čistého HTML struktury a obsahu může radikálně zvýšit použitelnost obsahu (např. čtení nevidomými) nebo sdílení CSS souboru mezi více stránkami a různé podobné výhody. Dále CSS umožňuje možnost nastavit různá stylování pro různá specifická zařízení. Kaskádové styly také specifikují prioritu aplikující se v případě kdy více než jedno pravidlo odpovídá. V případě splnění více pravidel se upřednostňuje to které přesněji odpovídá stromové (kaskádové) struktuře v které se prvek nachází – od toho slovo „kaskádové“. Soubor stylů s příponou `.css` obsahuje zadané definice pravidel. Pravidla obsahují *selektor* a *deklarační část*. Příklad použití a obsahu CSS souboru:

```
selektor {
	deklarační část
}
p, div {
	color: red;
	background-color: #fff;
}
```

Toto pravidlo má jako selektor prvek `<p>` představující paragraf a `<div>` blok. V deklarační části najdeme nastavení barvy textu na červenou (je možno použít i určité anglické slovo jako „*black*“ nebo „*red*“ apod. reprezentující barvu) a barvu pozadí ve zkráceném šestnáctkovém RGB formátu představující bílou barvu. Téměř každý prvek ve struktuře HTML lze vizuálně naformátovat pomocí CSS stylů. Pro vhodnější stylování lze využít i dvou obecných vlastností, které si nese každý prvek. A to identifikátor `id` a třídu `class`. Pomocí těchto dvojic můžeme také měnit styly HTML prvků. Příklad použití `id`:

```
<ul>
	<li>nedůležitý text</li>
	<li id="thisIsImportant">důležitý text</li>
	<li>nedůležitý text</li>
</ul>
<style>
	#thisIsImportant {color: red;}
</style>
```

Identifikátor `id` je jedinečným identifikátorem prvku a nesmí se dle specifikace vyskytovat u jiných prvků v dokumentu. Nyní příklad použití na třídy:

```
<ul>
	<li class="modry">nedůležitý text</li>
	<li class="modry">další text</li>
	<li class="modry">ještě další text</li>
</ul>
<style>
	.modry {color: blue;}
</style>
```

Pod definice třídy `class` může spadat i více jak jeden prvek. Na rozdíl od identifikátoru `id`, který musí být unikátní. Třída umožňuje definovat všechny prvky, které spadají do určitě třídy, a od identifikátoru `id` jsou naopak vhodnější pro definování skupiny prvků mající určitý společný význam.

3.4.2 CSS3

Kaskádové styly CSS3 jsou další generací kaskádových stylů a jsou rozděleny do několika dokumentů, které jsou v různých stádiích vývoje. Rozdělení na jednotlivé části společně s prioritou vývoje společně se stavem vývoje se nachází v Tabulka 3.4 – Aktuální stav jednotlivých částí CSS3 [8] a [9]. [8] a [9] níže.

Vysoká priorita:			
CSS Namespaces	CR		
CSS Backgrounds and Borders	CR		
CSS Multi-column Layout	CR		
Media Queries	CR		
Střední priorita:			
CSS Mobile Profile 2.0	CR	CSS Speech	WD
CSS Marquee	CR	CSS Basic User Interface	CR
CSS Paged Media	LC	CSS Scoping	-
CSS Print Profile	LC	CSS Grid Positioning	WD
CSS Values and Units	WD	CSS Grid Layout	WD
CSS Cascading and Inheritance	WD	CSS Regions	WD
CSS Text	WD	CSS Flexible Box Layout	WD
CSS Writing Modes	WD	CSS Image Values	WD
CSS Line Grid	-	CSS 2D Transformations	WD
CSS Ruby	CR	CSS 3D Transformations	WD

CSS Generated Content for Paged Media	WD	CSS Transitions	WD
CSS Fonts	WD	CSS Animations	WD
CSS Basic Box Model	WD	CSS style Attribute Syntax	CR
CSS Template Layout	WD		
Nízká priorita:			
CSSOM View	WD	CSS Hyperlink Presentation	WD
CSS Extended Box Model	-	CSS Math	-
CSS Object Model	WD	CSS Presentation Levels	WD
CSS Syntax	WD	CSS Aural Style Sheets	-
CSS Lists	WD	CSS TV Profile 1.0	CR
CSS Tables	-	Behavioral Extensions to CSS	WD
CSS Reader Media Type	WD	CSS Introduction	WD
CSS Positioning	-		
CSS Generated and Replaced Content	WD		
CSS Line Layout	WD		

Vysvětlivky[21]: WD = Working Draft, LC = Last Call, CR = Candidate Recommendation

Tabulka 3.4 – Aktuální stav jednotlivých částí CSS3 [8] a [9].

CSS3 se zaměřuje většinou na vizuální formátování, které šlo doposud těžce dosáhnout jako například stíny, průhlednost nebo podpora vložení vlastního formátu písma či rozsáhle typografické prvky. Umožňuje také visuální transformace a animace. Ale také několik starších vlastností ze starších verzí.

3.4.3 Animace

Animace dosažitelné pomocí CSS3 jsou zajímavou možností, které nebylo doposud možné s CSS2 dosáhnout. Vlastnost `@keyframes` specifikuje animaci ve tvaru například:

```
@keyframes nazev_animace {
      from {top:0px;}
      to {top:200px;}
}
```

Vlastnost `animation-name` přiřadí animaci k určitému prvku/prvkům dle názvu animace zadaným pomocí `@keyframes`. Vlastnost `animation-duration` nastaví čas (v sekundách) trvání animace.

[21] Použito názvosloví z kapitoly 2.2 na straně č. 5.

Vlastnost `animation-timing-function` tok času animace. Při zadání `linear` bude čas animace ubíhat lineárně u `ease` bude čas ubíhat pomaleji na začátku a na konci animace, uprostřed poběží rychleji. Podobně u zadání `ease-in` animace bude mít jen pomalý tok na startu u `ease-out` bude mít animace pomalý tok času na konci. Lze zadat i specifické chování dle bézierovy kubiky `cubic-bezier(n,n,n,n)` kde číslo `n` je v rozsahu od 0 do 1. Příklad:

```
div {
	animation-name: nazev_animace;
	animation-duration: 1s;
	animation-timing-function: linear;
	animation-delay: 2s;
	animation-iteration-count: 3;
	animation-direction: alternate;
	animation-play-state: paused;
}
```

Celý zápis lze zapsat i ve zkrácené podobě díky vlastnosti `animation: name duration timing-function delay iteration-count direction;` respektive `animation: nazev_animace 1s linear 2s 3 alternate paused;`

3.4.4 Okraje

Deklarování okraje prvky za pomocí `border` je hojně využíváno a bylo stanoveno již v CSS1. CSS3 přináší možnost zaoblit rohy takto vytvořeného ohraničení tvaru obdélníku či čtverce a to s využitím `border-radius` a nastavením požadovaného poloměru zaoblení všech rohů nebo při potřebě zaoblení specifikovaného rohu využitím `border-bottom-left-radius`, `border-bottom-right-radius`, `border-top-left-radius` či `border-top-right-radius`. Nově lze také využít možnost vytvoření okraje za pomocí obrázku a to pomocí vlastnosti `border-image: source slice width outset repeat` nebo také možnost určit okraji stín vlastností `box-shadow: h-shadow v-shadow blur spread color inset` v příkladu níže je nastaveno odsazení stínu v ose X i v ose Y `10px`, rozostření `5px` a barva stínu `#888888`:

```
box-shadow: 10px 10px 5px #888888;
```

3.4.5 Písmo

Doposud se museli programátoři webových aplikací spoléhat při stylování písma na přítomnost jisté sady písma obsažené v operačním systému na kterém uživatelům webový prohlížeč byl spuštěn. Nyní s CSS3 mohou naimportovat vlastní libovolná písma a to díky vlastnosti `@font-face` se zadává název souboru s potřebným písmem. Příklad použití:

```
@font-face {
      font-family: novePismo;
      src: url('Saation_Light.ttf'),
           url('Saation_Light.eot'); /* IE */
}
```

Pak již stačí toto tzv. naimportované písmo použít jako jakékoliv jiné:

```
div {
      font-family: novePismo;
}
```

3.4.6 Sloupce textu

Kaskádové styly CSS3 umožňuji nové možnosti zobrazení textu. Například vlastnost `column-count` určuje počet sloupců na které se rozdělí text obsahující prvek jež má takto nastaven styl. Vlastnost `column-gap` určuje volný prostor mezi sloupci. Podobně i vlastnost `column-rule` určuje obsah v mezeře mezi sloupci. V tomto případě oddělující svislou čáru. Vlastnost `column-span: all;` ruší zalomování textu do sloupců což je vhodné například pro nadpisy v blocích mající `column-count` vlastnost. Nastavením `column-width` specifikujeme šířku sloupců. Zkrácený zápis nejpoužívanějších vlastnosti tohoto nastavování je `columns: column-width column-count`.

3.4.7 Text

Specifikace CSS3 umožňuje podobně jako například u vlastnosti `border` nastavit stín i u textu. A to pomocí `text-shadow: h-shadow v-shadow blur color` kde `h-shadow` a `x-shadow` znamená odsazení stínu v horizontální respektive ve vertikální rovině. Hodnota `blur` znamená sílu rozmazání stínu a na místo `color` patří určení barvy stínu. Nové kaskádové styly umožňují povolení zalomení dlouhých slov na nový řádek za pomocí vlastnosti `word-wrap`.

3.4.8 Přechody

Nové specifikace CSS3 umožňuje řadu efektů, které umožňují animovat změnu mezi dvěma různými stavy/vlastnostmi určitého prvku. Například za pomoci `transition: property duration timing-function delay` kde `property` představuje vlastnost prvku, která se bude při změně animovat (měnit plynule), `duration` je čas délky této animace v sekundách, `timing-function` může nabývat hodnot stejně jako u nastavení animací na straně 29 a tedy umožňuje nastavit tok času animace. A hodnota `delay` specifikuje čas zpoždění. Ukázka použití:

```
div {
	width: 100px;
	transition: width 2s;
}
div:hover {
	width: 300px;
}
```

Zde bude prvek bloku `<div>` plynule měnit svou šířku po najetí myši a to z šířky 100px na 300px. Doba animace bude 2 sekundy.

3.4.9 Uživatelské prvky

CSS3 přináší i vlastnosti, které mají za úkol zjednodušit tvorbu uživatelsky příjemných prostředí. První vlastností, která má zjednodušit tvorbu prvků vizuálně podobných formulářovým je `appearance: normal|icon|window|button|menu|field` jež umožňuje vytvářet z prvku `<div>` malý obrázek (`icon`), tlačítko (`button`), soubor možností s kterých si může uživatel vybrat (`menu`) nebo textový vstupní prvek (`field`). Vlastnost `box-sizing` umožňuje definovat určité elementy tak aby se přizpůsobily určité oblasti určitým způsobem. Například pokud potřebujeme vytvořit dva boxy vedle sebe tak zvolíme vlastnost `box-sizing` a nastavíme ji na `border-box`:

```
div {
	box-sizing: border-box;
	float: left;
	width: 50%;
}
```

Pokud je potřeba aby elementům `<div>` mohla být uživatelem měněna velikost je možno z CSS3 využít vlastnost `resize: both` povolující změnu velikosti v obou (`both`) osách. V případě potřeby

změny jen v ose X nebo Y jo možno tohoto dosáhnout za pomocí `resize: horizontal` respektive `resize: vertical`.

3.4.10 CSS3 selektory

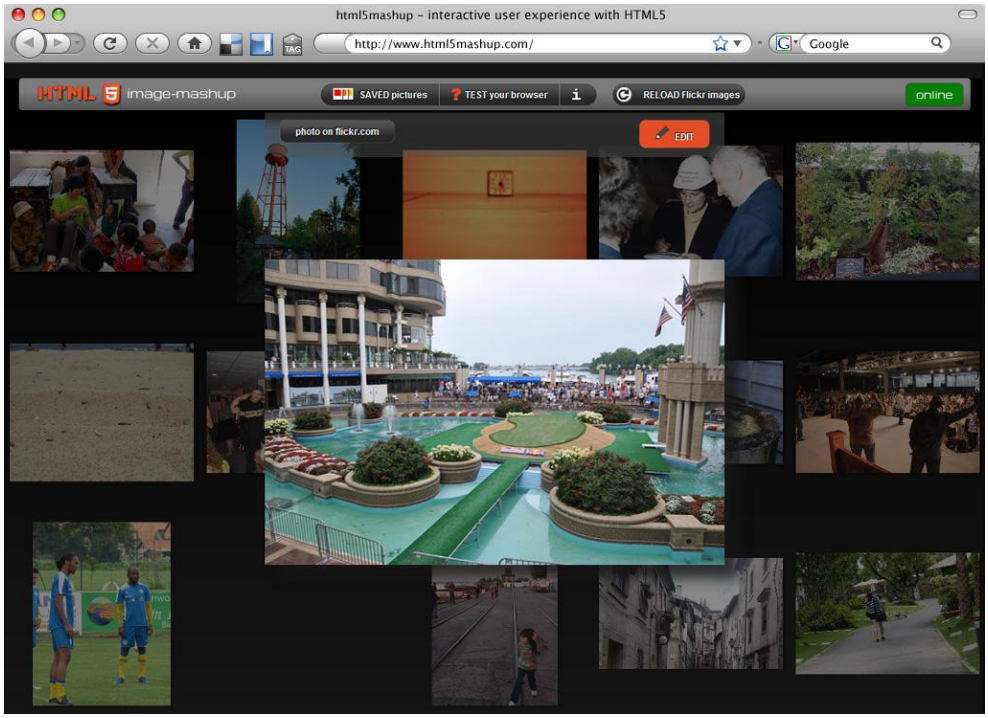
Pojem „*selektor*“ v CSS byl již zmíněn při základním pojednávání o CSS na straně 27. Pro připomenutí selektor je pravidlo, které odpovídá 0 až n prvků v HTML dokumentu. Selektor `*` označuje všechny elementy (prvky). Selektor `div, p` odpovídá všem prvkům typu `<div>` a `<p>`. Specifikace CSS3 přináší nové selektory, které zvětší možnosti při zadávání specifických pravidel. Selektor `prvek1~prvek2` znamená výběr každého `prvku1`, který je předcházen `prvkem2`. Selektor `[atribut^=hodnota]` znamená výběr prvku, který má `atrbut` a jeho začínající jako `hodnota`. Pro příklad `a[src^="https"]` vybere všechny prvky `<a>` s parametrem `src` jež začíná „`https`“ řetězcem. Podobně lze vybrat i parametry končící nebo obsahující určitý řetězec za pomocí `$=` respektive `*=`. Selektor `:first-of-type` vybere veškeré prvky jež jsou prvními prvky jejich rodičů. Podobně lze použít `:last-of-type` lze vybrat poslední prvky a u `:only-of-type` lze vybrat prvky které jsou samotnými u jejich rodičů. Selektor `:nth-child(n)` vybere každý prvek, které je n-tý v jeho rodiči. Selektor `:root` vybere kořenové element. Selektor `:empty` vybere prvky, které nemají žádné dítě (*child element*). Selektor `:enable`, `:disable` a `:checked` vybere všechny povolené vstupní elementy respektive všechny zakázané vstupní elementy respektive všechny zatržené vstupní elementy. Negace v selektoru se vytváří pomocí `not()` a tedy `not(p)` vybere veškeré prvky, které nejsou prvky `<p>`.

4 Demonstrační webová aplikace

Tato kapitola popisuje celou implementaci demonstrační webové aplikace rozdělenou na jednotlivé funkční a typově oddělitelné části. Čtenář se postupně dozví o využití všech prvků z HTML5 a CSS3. Od využití CSS3 možností vizuální tvorby a animací webu přes využití plátna *canvas* na editování obrázků dále o ukládání obrázků do *Lokální paměti* a až po využití režimu *offline*. Následující text stručně popisuje funkčnost, kterou výsledná webová aplikace nabízí.

4.1 Popis webové aplikace

Jedná se o webovou aplikaci prohlížeč obrázků s názvem „*HTML5 image-mashup*“ demonstrující výhody standardů HTML5 a CSS3 a skriptovacího jazyka *JavaScript* společně s implementací PHP na straně serveru. Aplikace jako zdroj dat využívá nejnovějších veřejných fotografií z internetové databáze *www.flickr.com* pomocí PHP *Flickr API* knihovny jména *PEAR::Flickr_API* [10]. Tyto fotky jsou zobrazovány pomocí CSS transformací, animací a dalších efektů. Vybrané obrázky může uživatel následně editovat a uložit a tím si vytvořit poznámku k pozdějšímu zobrazení. K uložení slouží *Lokální paměť* webového prohlížeče. Její základní výhodou je vlastnost, že v ní uživatelem uložené obrázky zůstanou uloženy i po uzavření prohlížeče na neomezeně dlouhou dobu. V *offline* režimu celá aplikace funguje omezeně a logicky nemůže načítat nové obrázky z online databáze *Flickr*. Uživatel má ovšem přístup k poznamenaným, uloženým obrázkům a má možnost si je prohlížet a i zpětně editovat a opětovně uložit. Nad uloženými obrázky má uživatel také kontrolu v představě mazání jednotlivých položek společně s možností vymazat celou *Lokální paměť*. Indikace *offline* či *online* režimu se nachází v pravé části nahoře umístěné hlavní lišty (oObrázek 4.1 – Náhled webové aplikace v prohlížeči Google Chrome 12.0.747). Aplikace obsahuje i jednoduchý test HTML5 podpory. Grafika celého uživatelského prostředí je navržena jako jednoduchá, přehledná a uživatelsky příjemná.



Obrázek 4.1 – Náhled webové aplikace v prohlížeči Google Chrome 12.0.747

4.2 Struktura

Webová aplikace je postavena na prvcích HTML5 zdůrazňující sémantiku obsahu (kapitola 3.1.1 a kapitola 3.1.2). Na prvním řádku je tedy HTML5 `<!DOCTYPE>`, jenž nám tímto definuje HTML5 typ dokumentu. Podobně tak zkrácená je i HTML5 verze kořenového prvků `<html>`, kde je definován jazyk `lang="en"` v jeho vlastnostech a soubor nastavující a specifikující *offline* režim `manifest="/cache.manifest"` o tomto ale až později (v kapitole 4.6). Dále v prvku `<head>` najdeme HTML5 zápis vložení generovaných kaskádových stylů za pomocí prvku `<style>` a kódu skriptovacího jazyka `<script>`. Přestože tato webová aplikace neposkytuje velké množství informací, které by se mohly sémanticky zvýraznit za pomocí HTML5 prvků, tak i přesto bylo využito několik takovýchto prvků. Jako například u horní lišty pomocí prvku `<menu>` nebo u výčtu obrázků prvkem `<section>` či u zobrazení uložených obrázku prvkem `<aside>`.

Prvky HTML5, které rozšiřují sémantiku oproti HTML4 je mnoho. Veškeré tyto speciální prvky lze nahradit HTML4 prvky `<div>`, `<span>` apod. jen jejich sémantika nebude nikdy jednoznačná.

4.3 Vizuální podoba

U této webové aplikace byl využit jen jeden sdílený CSS `style.css` soubor pro veškeré HTML soubory. Ostatní CSS obsah je generován do `<style>` prvku. Tento soubor je logicky stavěn a podobná pravidla jsou napsána v pořadí za sebou tak, aby bylo možné jednodušší případné vyhledávání. Snahou bylo využít potenciál vizuálních efektů CSS3 a nevyužít tak téměř žádné obrázky apod. „dodělky“, které se v hojné míře využívají. Podobné techniky jsou ale v dnešním světě webu téměř nutností pro zachování zpětné kompatibility se starými prohlížeči při zachování vizuálně nejpodobnějšího výsledku. Bohužel tímto ale zvětšují mnohdy několikanásobně objem přenesených dat při načítání. Tuto situaci pomohlo výrazně zlepšit v případě této webové aplikace právě CSS3.

Nyní se zaměříme na konkrétní efekty a možnosti, které byly ve webové aplikaci využity. Začněme například stíny. Ať už textu deklarací `text-shadow`, tak bloku prvku `<div>`, na obrázcích `<img>` či hypertextových odkazech `<a>` pomocí vlastnosti `box-shadow`. Jelikož se jedná o experimentální vlastnost, bylo využito i speciálních prefixových vlastností pro zvýšení podpory mezi prohlížeči jako `-moz-box-shadow` uplatňující se v prohlížečích *Mozilla Firefox* a `-webkit-box-shadow` pro prohlížeč s jádrem *Webkit* jako je *Google Chrome* (či jeho vývojovou verzi *Chromium*) a *Safari*. Bohužel prohlížeč *Opera* prozatím nepodporuje tento druh efektu. Co se týče podpory stínu u prohlížeče *Windows Internet Explorer*, jen verze 9 obsahuje tuto podporu [11]. V případě stínů bylo také využito možnosti definice barvy s průhledností v RGBA modelu. Kde zkratka RGBA znamená barvy Red, Green a Blue a tzv. Alfa kanál definující průhlednost. Například: `rgba(255, 255, 255, 0.5)` znamená maximální hodnotu červené, zelené i modré (ve výsledku bílá) a poslední hodnotou je alfa kanál od 1 do 0, kde 1 je stoprocentní viditelnost (neprůhlednost).

Jako další deklarací vlastnosti je zaoblení krajů `border-radius`, který definuje zaoblení okraje o zadaném poloměru. V případě zaoblení určitého rohů je využito `border-bottom-left-radius` deklarace a její mutace. Tuto vlastnost podporují všechny testované majoritní webové prohlížeče.

Co se týče gradientního pozadí použitého na pozadí za maticí obrázků a v hlavní liště i na tlačítcích, bylo pro různé prohlížeče dosáhnuto různými způsoby s využitím speciálních prefixů podobně jako výše u vlastnosti `box-shadow`. Opět se to jedná spíše o experimentální vlastnost a způsob zadání vlastnosti tomu napovídá. U prohlížečů s jádrem *Webkit*, je deklarace následující:

```
background:-webkit-gradient()
```

Podobně je tomu i pro Mozilla Firefox a Opera:

```
background: -moz-linear-gradient()
background: -o-linear-gradient()
```

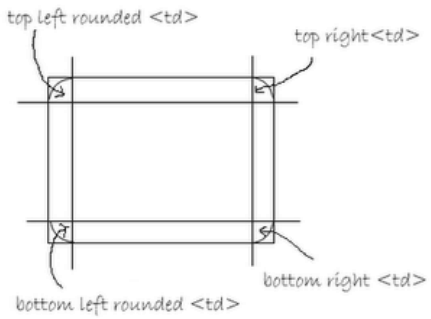
Internet Explorer potřebuje naprosto jiný přístup. Je zde potřeba nadefinovat speciální filtr:

```
filter: progid:DXImageTransform.Microsoft.gradient()
```

Využito bylo i možnosti definování průhlednosti obrázku či blok prvku pomocí vlastnosti `opacity` ve stupnici opět od 0 do 1, kde 1 je neprůhlednost. Toto je například využito při zvětšení vybraného obrázku kliknutí myši, kde se ostatní obrázky právě díky průhlednosti ztmaví (jelikož pozadí je tmavé) a nenarušují prohlížení vybraného obrázku.

Při zvětšování či nastavování měřítka obrázku bylo také využito možností CSS3. Tato možnost umožnila vytvoření velmi uživatelsky příjemného prostředí funkcí, kterých nelze za pomocí starších (aktuálních) standardů dosáhnout. Transformace a animace jsou majoritně využity při polohování a změně velikosti obrázků v zobrazení na hlavní obrazovce a v náhledu uložených obrázků. Využitím `-webkit-transform: translate(x,y)` bylo dosaženo posunutí objektu v ose `x` i `y` dle zadaných hodnot (pixelů) uvnitř závorek. Další využitá transformace je `-webkit transform: scale(0.5)`, jenž umožňuje prvek zmenšit a to v tomto případu na poloviční velikost. U `-webkit-transform-origin(x,y)` je možno přenastavit výchozí bod odkud se transformace provádí. Výpočet optimální pozice obrázku ve webové aplikaci se provádí pomocí rozlišení zobrazovacího okna prohlížeče a velikosti obrázku s ohledem na dodržení zadaných odstupu mezi sebou. Jelikož se opět jedná o experimentální vlastnost bylo nutné aplikovat opět speciální prefixové vlastnosti pro majoritní prohlížeče jako `-webkit-transform`, `-moz-transform` a `-o-transform`.

Pokud porovnáme tyto možnosti s možnosti, které doposud CSS2.1 přinášelo tak lze s jistotou říct, že tento standard žádné obdobné možnosti vytvoření animací, transformací, stínů nebo zaoblení okrajů doposud neposkytoval. Jedinou možností jak grafických prvků jako jsou stíny a zaoblení okrajů docílit je za pomocí doplňujících obrázků a jejím vhodným polohováním (jako například na Obrázek 4.2). Je ovšem nutné využít pokročilejšího grafického editoru pro tvorbu efektu a také objem přenesených dat se mnohonásobně zvětší oproti použití vlastností CSS3.



Obrázek 4.2 – Jedna z metod vytvoření zaoblených okrajů pomocí prvků `<div>` bez CSS3. Obrázek převzat z [12].

Jednodušších efektů lze také dosáhnout technologií *JavaScript*, která dokáže přistupovat a pracovat s DOM reprezentací dokumentu společně s využitím aktuálně doporučovaných standardů HTML4 a CSS2.1 a vytvořit funkce napodobující animace apod. Na tuto problematiku se krom jiných zaměřuje *JavaScriptová* knihovna *jQuery* [13], která při využití předpřipravených funkcí tuto práci usnadňuje. Pokud by bylo využito technologie *Flash* tak veškeré výše zmíněné vizuální efekty CSS3 by byly plně nahraditelné. Ovšem s několika nevýhodami. *Flash* je doplněk pro webové prohlížeče firmy Adobe a tudíž nebude zaručena jeho přítomnost v prohlížeči. Samotný vývoj je náročnější než využití vlastnosti CSS3 a navíc vývojové prostředí jako „*Adobe Flash Professional CS5*" pro tvorbu *Flash* animací a aplikace patří mezi dražší, konkrétně okolo hranice 850€ z oficiálního obchodu Adobe. Oproti tomu vývojové prostředí pro tvorbu webových aplikací HTML5 a CSS3, kde lze využít jakéhokoliv textového editoru, nemusí stát programátora vůbec nic.

Podpora CSS3 v majoritních prohlížečích je znázorněna v následující Tabulka 4.1 kde lze vidět, převaha podpory u prohlížečů s jádrem *Webkit* (*Chromium* a *Safari*).

Internetový prohlížeč	border-radius	opacity	rgba()	hsla()	box-shadow	text-shadow
Chromium 12.0.747	ano	ano	ano	ano	ano	ano
Firefox 4.0.1	ano	ano	ano	ano	ano	ano
Opera 11.50	ano	ano	ano	ano	ano	ano
Internet Explorer 9.0.8112	ano	ano	ano	ano	ano	ne
Safari 5.0.5	ano	ano	ano	ano	ano	ano
Internetový prohlížeč	Background Size	Multiple backgrounds	border-image	CSS Animations	CSS Columns	CSS Gradients
Chromium 12.0.747	ano	ano	ano	ano	ano	ano
Firefox 4.0.1	ano	ano	ano	ne	ano	ano
Opera 11.50	ano	ano	ano	ne	ano	ano
Internet Explorer 9.0.8112	ano	ano	ne	ne	ne	ne
Safari 5.0.5	ano	ano	ano	ano	ano	ano
Internetový prohlížeč	CSS Reflections	CSS 2D Transforms	CSS 3D Transforms	CSS Transitions	FlexBox	
Chromium 12.0.747	ano	ano	ano	ano	ano	
Firefox 4.0.1	ne	ano	ne	ano	ano	
Opera 11.50	ne	ano	ne	ano	ne	

Internet Explorer 9.0.8112	ne	ano	ne	ne	ne
Safari 5.0.5	ano	ano	ano	ano	ano

Tabulka 4.1 – Podpora CSS3 vlastností v majoritních prohlížečích.

4.4 Canvas

Prvku `<canvas>` je tedy využito pro práci s obrázkem konkrétně při jeho editaci po stisknutí tlačítka 

. Tímto se provede přesměrování prohlížeče na stránku `/canvas/index.php`, kde se vybraný obrázek „nahraje“ pro úpravu do plátna `canvas`. Zde je možno kreslit pomocí funkce `draw()` a otáčet pomocí funkce `rotate()` kde je obsah plátna uložen, vymazán funkcí `clearCanvas()` a postupně plynule vykreslován postupným pootočením metodou `rotate()`. Obsah plátna `canvas` společně se zakreslenými poznámkami lze uložit jako obrázek typu `jpeg` (v případě formátu `png` je datová náročnost několika násobně větší[22]) a uložit jej do *Lokální paměti* (více v kapitole 4.7 o *Lokální paměti*) po převodu metodou `canvas.toDataURL("image/jpeg")` funkcí `save()`.

Inicializace plátna `canvas` probíhá pomocí `document.getElementById('myCanvas')` a s ním propojený 2D obsah `canvas.getContext('2d')`. Vložení obrázku provedeme pomocí metody `drawImage()` metody pro rotaci nebo `context.lineTo()` pro posunutí kreslícího štětce. Pro detekci pohybu a kliknutí myši na plátno byly využity tyto tři kontroly událostí popisující chování a vyvolání potřebné funkce při stisku tlačítka myši dolů či nahoru nebo při pohybu myši:

```
canvas.addEventListener('mousedown', ev_canvas, false);
canvas.addEventListener('mousemove', ev_canvas, false);
canvas.addEventListener('mouseup',   ev_canvas, false);
```

Při implementaci plátna `canvas` společně s využitím *flickr API* nastal problém. Jelikož webová aplikace je postavena na datech, která získává nikoli z vlastního úložiště, ale ze vzdáleného serveru *Flickr.com* a nahrání obrázku na plátno, který není lokálně uložen (nenachází se na stejném serveru jako aplikace) není z bezpečnostního hlediska možný. Tento problém je vyřešen krátkodobým přechodným uložení dat (aktuálně editovaného obrázku) na straně PHP serveru. Nyní něco o identifikaci konkrétního uživatele. Každý uživatel nebo přesněji webový prohlížeč je na straně serveru identifikován určitým jedinečným *ID Session* a to za pomocí tzv. *Sessions* (česky sezení), které PHP k tomuto využívá. Toto sezení po určité době neaktivity vyprší a zároveň se vymažou i veškerá dočasná data jako poslední příchozí XML data s daty z *Flickr* databáze (více v kapitole 4.5 Flickr API). Vybraný soubor, který je tedy potřeba editovat tedy stáhneme do dočasné složky na

[22] Záleží na konkrétním obsahu plátna.

server. Využijeme přitom knihovny `libcurl` a možností `cURL`, která umožňuje komunikaci serveru s jiným serverem. Takto uložený soubor je vždy pojmenován stejně a to jako *ID Session* klienta, který si o obrázek žádá. Tudíž počet souborů odpovídá počtu aktivních *Sessions*. A pomocí kontroly vyvolané na začátku každého požadavku o uložení nového souboru, je zjištěno zda veškeré soubory uložené odpovídají seznamu aktivních *Sessions*. V případě jestli nějaká *Session* vypršela a stane se neaktivní, tak se i vymaže soubor s ní spřažený.

Ve starším standardu HTML4 se nenachází nic podobně využitelného jako je právě prvek `<canvas>`. Jako alternativu by šlo opět využít technologie *Flash* ale s opět stejnými zápornými vlastnostmi, který byly zmíněny na konci kapitoly 4.3 u tvorby vizuální podoby. Využít by bylo možné taky technologie *JavaScript* a starších standardů HTML4 a CSS2.1, ale s nejistým a zřejmě náročným úspěchem. Oproti tomu CSS3 a *JavaScript* nabízí relativně jednoduché metody jak výsledku dosáhnout.

Podpora majoritních prohlížečů je zobrazena níže v Tabulka 4.2 kde lze vidět, že je tento prvek podporován ve všech majoritních prohlížečích. Ovšem podpora je v mnoha případech neúplná a implementace standardu se mnohdy liší. Například získané souřadnice aktuální polohy myši se mohou lišit. Toto jsou poznatky, které jsem získal při implementaci a testování aplikace a nepodařilo se mi nalézt chybu v mé implementaci webové aplikace.

Internetový prohlížeč	Canvas	Canvas Text
Chromium 12.0.747	ano	ano
Firefox 4.0.1	ano	ano
Opera 11.50	ano	ano
Internet Explorer 9.0.8112	ano	ano
Safari 5.0.5	ano	ano

Tabulka 4.2 – Podpora `canvas` prvků v majoritních webových prohlížečích.

4.5 Flickr API

Jako zdroj dat – fotografií je využito veřejných fotografií z databáze *Flickr*. Pro jejich načítání je využíváno *Flick API* [14] pro PHP. Jedná se o soubor metod napomáhající získání potřebných fotografií, které jsou využity jako zobrazovaný obsah. Implementace se nalézá v soboru `core.php` a

využívá tohoto *API*. Pro používání tohoto *API* je potřeba se zaregistrovat na *The App Garden*[23] ve službách *Flickr* a uvést účel a cíl použití pro které *Flickr API* hodláte použít. Poté použijeme registraci získaný klíč v *API*. Toto *API* využívá *PEAR framework*, který řeší odlišný přístup PHP k databázím, logováni či autentizaci. Tento *framework* je potřebný k běhu *API* a byl potřeba doinstalovat. Po nastavení proměnné klíče `key` a vložení všech potřebných souborů nic nebrání v navázání spojení. Spojení otestujeme následovně:

```
$api->callMethod('flickr.test.echo', array( 'foo' => 'bar', ));
```

Tato funkce nám tedy otestuje připojení a pokud se spojení nevydařilo, tak číslo chyby a její popisek bude získán z `$api->getErrorCode()` a `$api->getErrorMessage()`. V tuto chvíli již lze požádat o potřebná skutečná XML data metodou pomocí které, dostaneme nově přidané fotky:

```
$api->callMethod('flickr.photos.getRecent', ... );
```

Odpověď je nám vrácena v objektu v podobě XML_Tree[24] a obsahuje data která potřebujeme. Je tedy nezbytné tuto odpověď rozparsovat. Z XML dostaneme několik malých dílů informací, které vhodným spojíme za sebou vytvoříme validní URL adresu obrázku. Tuto adresu opakovaně přidáváme do proměnné `$out` z které se bude později vypisovat část HTML kódu. Takto se tedy skládá výsledná adresa obrázku:

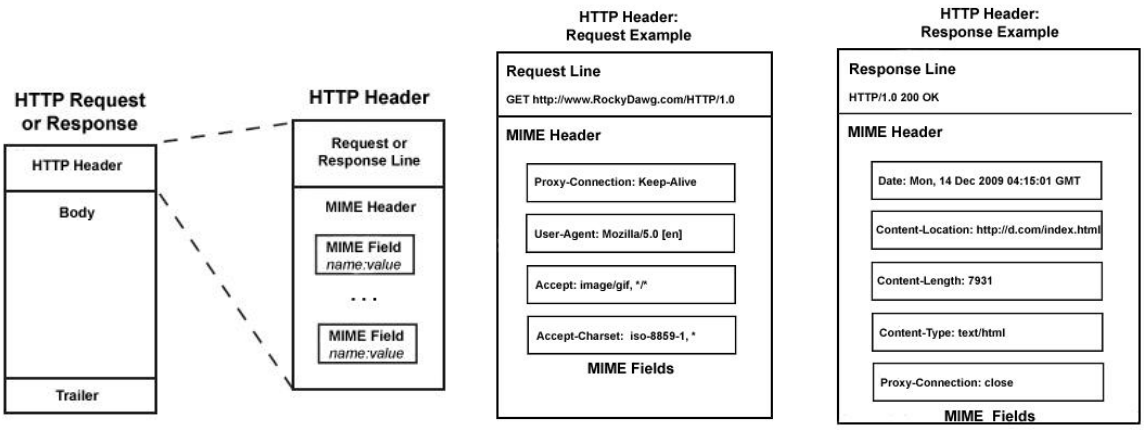
[23] Sklad různých API využívající databázi obrázku *Flickr* společně se soupiskou stránek tyto API využívající.
[24] Ukázková odpověď stromu XML_Tree se nachází v příloze č. 1

4.6 Offline

Webová aplikace umožňuje chod a funkčnost i v režimu *offline* neboli bez připojení k internetu. Pokud aplikace detekuje odpojení od internetu, tak přejde do omezeného *offline* režimu, kde je umožněno přistupovat k uloženým obrázkům a pracovat a ukládat je jako v normálním režimu. Toto je řešeno za pomocí tzv. `manifest` souboru jež určuje, které soubory budou uloženy do aplikační cache pro případné pozdější využití v *offline*. Umístění `manifest` souboru je určeno v kořenovém prvku `<html>`, a to přes vlastnost `manifest`, a to následovně:

```
<html manifest="/cache.manifest">
```

Takto prohlížeč pozná, že má načíst `chache.manifest` soubor z kořenového adresáře. Po načtení se automaticky začnou načítat i všechny soubory jež se mají načíst do aplikační cache prohlížeče. Dle standardu je prohlížeč schopen manifest soubor načíst pouze pokud se jeví jako soubor MIME[25] typu „*application/manifest*“. Každý požadavek na server, například o stažení určitého souboru, je posílán jako tzv. *HTTP Request*, a taktéž server zpět odpovídá klientovi pomocí této zprávy HTTP protokolu. Identifikace, o jaký soubor se při přenosu jedná, najdeme v HTTP hlavičce v části *Content-Type* (Obrázek 4.3 – Znázorňuje umístění HTTP hlavičky v HTTP Request/Response společně s obsahem HTTP Header.).



Obrázek 4.3 – Znázorňuje umístění HTTP hlavičky v HTTP Request/Response společně s obsahem HTTP Header. Obrázek převzat z [15].

Soubor `manifest` tedy musí mít v hlavičce vráceného požadavku *Content-Type* zadáno *application/manifest*. Toto rozpoznání souborů probíhá na straně serveru a může být někdy

[25] Multipurpose Internet Mail Extensions byl původně navržen jako víceúčelová rozšíření internetové pošty, ale nyní jej používají i jiné internetové protokoly jako např. HTTP.

problémem, jelikož `manifest` soubory nejsou v standardních konfiguracích rozpoznávány. Tohoto dodatečného nastavení lze docílit dvěma způsoby. Prvním je manuálně přidat v nastavení serveru rozpoznávání *MIME* typů podle potřebné přípony. Druhým je nastavení skrze soubor `.htaccess`. Právě tento druhý způsob byl využit u testování, jelikož testovací prostor se nachází na hosting serveru, který nevlastním a tudíž nemám ani přístup do jeho interního PHP nastavení. Proto bylo využito možnosti druhé a vystačilo přidat v kořenovém adresáři soubor `.htaccess` s novým řádem „`AddType text/cache-manifest .manifest`" přiřazující příponu a rozpoznání k námi požadovanému typu souboru. Pro kontrolu detekce typu souboru z HTTP hlavičky je možno využít webovou stránku *web-sniffer* [16] jež umožňuje zobrazit po zadání cesty požadavku HTTP hlavičku odpovědi. Níže bude zobrazen obsah použitého `manifest` souboru a vysvětleny funkce a jeho vnitřních částí:

```
CACHE MANIFEST
# Cached files to offline run

CACHE:
offline.html
style.css
img/html5.png
canvas/img/delete.png
canvas/img/erase.png
canvas/img/pencil.png
canvas/img/rotate.png
canvas/img/save.png
canvas/rotate.js
canvas/draw.js
Michroma-webfont.woff
Michroma-webfont.ttf
Michroma-webfont.svg

NETWORK:
*

FALLBACK:
/ /offline.html
```

První řádek je povinný. Prázdné řádky jsou ignorovány společně s řádky začínající `#` pro komentáře. Do části hned pod `CACHE` hlavičkou, jsou uvedeny soubory, které mají být uloženy pro potřeby *offline* běhu. V sekci `NETWORK` najdeme soubory, jež mají být ignorovány v načtení do cache. V tomto případě znak `*` znamená, že budou ignorovány všechny stránky (krom těch v sekci `cache`). Sekce `FALLBACK` má funkci nahrazení všech stránek v *offline* za stránku `offline.html`. Důležitým pojmem je `master entry`, což je jakýkoliv HTML soubor obsahující definici `manifest` souboru v `<html>` prvku. Tento HTML soubor je totiž automaticky načten do cache bez ohledu na obsah `manifest` souboru (jestli se nachází v sekci `NETWORK`). Toto z počátku implementace dělalo problém jelikož při generování nových obrázků z *Flickr* databáze je potřeba načíst nově vytvořený obsah `index.html` souboru – tedy soubor `master entry`. A toto systém *offline* přístupu k souborům dle specifikace

neumožňuje, jelikož tento soubor je `master entry` a tedy se již nachází v cache webového prohlížeče. Tento problém byl řešen následující způsobem: Při prvním navštívení stránky se nasměruje webový prohlížeč na soubor `index.html`, kde se i načte `manifest` soubor se všemi potřebnými soubory do cache. Poté při jakémkoliv dalším pokusu znovu vygenerovat a načíst nové fotografie se přesměruje na `master.html` soubor, jež je generován identicky jako `index.html`, jen nemá v prvku `<html>` definici `manifest`, a tudíž ani není `master entry`, a tedy ani nebude načten do cache a je ho možné načíst ze serveru s potřebným novým obsahem.

Pokud bychom chtěli *offline* režim nahradit s využitím jiného než HTML5 standardu tak jediný způsob je využitím Google a jeho *Gear API* [17] z než *offline* režimu v HTML5 vychází. Je nutné ovšem doinstalovat doplněk do prohlížeče. Dle zdroje [18] se vývoj *Google Gear* ovšem zastavuje a společnost Google se zaměřuje spíše na podporu HTML5 ve svých prohlížečích.

Co se týče podpory *offline* režimu v majoritních prohlížečích všechny webové prohlížeče kromě *Internet Exploreru 9* tento režim podporují.

4.7 Lokální paměť

Lokal Storage (česky *Lokální paměť*) je jedno ze dvou druhů uložišť typu *Web Storage* (podrobněji popsáno v kapitole 3.2.5) a stručně řečeno umožňuje ukládání dat i mezi okny. Na rozdíl od *Session Storage*, kde se data po uzavření okna mažou. V implementaci je tato možnost využita k uložení upravených obrázku pro pozdější zobrazení či editaci. A to i v případě když dojde k uzavření webového prohlížeče tak data stále zůstanou uložena. Ukládání dat do *Lokálního úložiště* se děje pomocí jednoduché metody `localStorage.setItem(klíč, data)`. Daty je v tomto konkrétním případu implementace myšlen obsah plátna. Nejprve je ovšem obsah celého plátna převeden na obrázek za pomocí metody `canvas.toDataURL("image/jpeg")` a až poté jej přiřadíme pomocí unikátního klíče do *Lokání paměti*. Z této paměti lze potřebnou položku vyzvednou metodou `localStorage.getItem(klíč)` jež patřičná odpovídající data vrátí. Tyto data poté poslouží jako zdroj dat konkrétního obrázku v prvku `<img>` přímo jako obsah parametru `src`. Při zobrazení všech fotografií[26] v *Lokální paměti* je využito standardního cyklu procházení polem se zarážkou průchodu. Zarážkou je hodnota `max` představující počet a zároveň název poslední uložené položky uložených obrázků v *Lokální uložišti*. Při mazání jednotlivých obrázků je využito metody s názvem `localStorage.removeItem(klíč)`, která umožní vymazání konkrétního záznamu (fotografie). Před provedením „*erase all*“ (česky „vymazat vše“) nejprve webový prohlížeč vyvolá vyskakovací okno upozorňující na mazání všech záznamů a po potvrzení se provede metoda `localStorage.clear()`, která kompletně vymaže *Lokálního úložiště*. Při používání *Lokální paměti* je možné narazit na maximální možnou hranici kapacity *Lokální paměti*, která je definována

[26] V editačním okně je vespod zobrazován náhled veškerých uložených obrázků.

samotným webovým prohlížečem. Obvykle se tato hodnota pohybuje okolo hranice 5MB. Díky využití možnosti ukládání obrázků do komprimované podoby `„image/jpg“`, které je několika násobně úspornější než-li `„image/png“`, lze do *Lokální paměti* uložit obvykle až 25 obrázků o rozlišení obrázku doručované skrze *Flickr API* s nejdelší stranou 500 pixelů.

Jedinou alternativou *Lokální paměti* v HTML5 (podle kapitoly 3.2.5) je využití *cookies*. Ty neumožňují ukládání velkého množství dat, a které jsou navíc přenášeny v každém HTTP požadavku. Možností je využití *Flash* technologie s již zmíněnými problémy nebo technologie Google Gear, která je ovšem na ústupu a bude ji plně nahrazovat právě HTML5 řešení.

Využívání *Lokální paměti* podporují veškeré majoritní prohlížeče i včetně statisticky nejvíce zaostávajícího *Internet Exploreru 9*.

4.8 Podpora prohlížečů

Všechny testy kompatibility byly prováděny na majoritních (pokud možno) nejaktuálnějších verzích webových prohlížečů. Konkrétně na *Chromium 12.0.747*, *Firefox 4.0.1*, *Opera 11.50*, *Internet Explorer 9.0.8112* a *Safari 5.0.5*. Veškeré srovnání podpory v předcházejících kapitolách využívalo testovacích výsledků z webové stránky *findmebyIP.com* [19]. Dle výsledné funkčnosti i z předchozích kapitol i z výsledků *findmebyIP.com* [19] lze usoudit, že webovými prohlížeči, které mají nejlepší podporu nových standardů HTML5 a CSS3 jsou prohlížeče disponující vykreslovacím jádrem *Webkit* a tedy *Google Chrome* (vývojová verze *Chromium*) a *Safari*. Pokud se zaměříme na reálné ohodnocení funkčnosti a reálné podpory tak prohlížeče s jádrem *Webkit* se zobrazením a funkčností stránek neměly žádný problém. *Internet Explorer* nepodporuje animace typu `transition` a definice určitých typu vizuálních efektů byly odlišné (pokud byly vůbec možné – podporované možnosti v Tabulka 4.1 – Podpora CSS3 vlastností v majoritních prohlížečích.). Prohlížeč *Opera* sice podporuje velké množství CSS3 vlastností i HTML5 prvků, ale například nedokázal načíst obrázek pro úpravu přestože prvek `<canvas>` podporuje. Prohlížeč *Firefox* měl problémy s kreslení na plátno *canvas* konkrétně souřadnice myši neodpovídaly skutečnosti. V chování prohlížeče *Chromium* jsem vypozoroval jisté nedeterministické chování konkrétně v CSS3 transformacích kdy se v jistých případech zobrazení obrázků na hlavní straně neprovedlo korektně – obrázky měly špatné pozice. Toto chování by se mohlo dát připsat ještě experimentálním vlastnostem CSS3, kterou transformace zajisté ještě je.

5 Závěr

Cílem této práce bylo nastudovat webové standardy HTML5 a CSS3 z hlediska rozšíření oproti svým předchůdcům, zmapování jejich podpory a na tomto základě vytvořit netriviální aplikaci. Komponenta, založená na nastudování těchto rozšíření možností oproti starším standardům, prověří a demonstruje realizaci interaktivních a vizuálních vlastností. Prvků rozšiřujících nové standardy je několik a jsou určeny pro mnohdy od sebe odlišné oblasti použití. Tyto prvky jsou také v odlišných stádiích vývoje. Proto jsem při implementaci webové aplikace kladl důraz na co nejlepší a nejvhodnější propojení nejdospělejších (z hlediska vývoje) prvků a částí ze specifikace HTML5 a CSS3. V kapitole zabývající se implementací jsou u každé podkapitoly, rozebírající určitou oblast možností nových standardů, také zmíněny konkrétní možnosti či nemožnosti řečení některých problémů, které by v rámci využití starších standardů mohly nastat společně s podporou na konkrétních webových prohlížečích.

Webové standardy HTML5 a CSS3, kterými se tato práce zabývá, jsou opravdu ještě ve stádiu návrh a různé stupně stádia návrhu a neúplná podpora webových prohlížečů tomu odpovídají. Ovšem určité prvky jsou použitelné a využitelné ve speciálních případech a aplikacích již dnes. Do budoucna se určitě stav standardu bude zlepšovat společně s ním i rozsah použitelnosti v prohlížečích.

5.1 Přínos projektu

Projekt (webová aplikace) ukazuje možnosti, které zvolené části vyvíjejících se moderních webových standardu již dnes přinášejí, společně s rozborem podpory webovými prohlížeči či případné problémy, které mohou programátora překvapit. Společně s tímto rozborem založeném na využití nejnovějších webových prohlížečů jsou porovnávány i možnosti řešení či nemožnost řešení se staršími standardy HTML4 a CSS2.1.

5.2 Možná rozšíření

Další zdroje fotografií

Webová aplikace umožňuje uživateli načítat obrázky z webové databáze *flickr.com*. Toto by se v budoucí verzi mohlo rozšířit a zavést možnost výběru mezi dalšími obdobnými webovými databázemi fotografií jako *Picasa Web Albums*[27] nebo *Smugmug API*[28].

[27] http://code.google.com/intl/cs-CZ/apis/picasaweb/overview.html
[28] http://wiki.smugmug.net/display/API/Home

Využití dalších prvků

V nynější době se nachází v pokročilejším stádiu vývoje jen určitá část z nových standardů a z této části je jen jistá část podporována webovými prohlížeči. Ovšem jelikož se jedná o standardy vyvíjející tak v budoucnu bude jak stádium vývoje tak i podpora v prohlížečích vylepšována. A bylo by možné zakomponovat i jistá další rozšíření do webové aplikace. Jako například využití *Geolocation API*.

Zobrazení fotografií

V budoucnu při zlepšení stavu standardu CSS3 a zdokonalení vlastností 3D transformací v něm obsažených by se mohla webová aplikace zdokonalit v náhledu fotografií a vytvořit například druhá možnost zobrazení v podobě 3D karuselu v prostoru.

A. Literatura

[1] JACOBS, Ian. *W3C Technical Report Development Process* [Online]. 2005 [cit. 2011-06-08]. Dostupné z: <http://www.w3.org/2005/10/Process-20051014/tr.html#q74>.

[2] SMITH, Michael. *section – section (NEW) - HTML5* [Online]. [cit. 2011-07-01]. Dostupné z: <http://dev.w3.org/html5/markup/section.html#section>.

[3] SMITH, Michael. *article – article (NEW) - HTML5* [Online]. [cit. 2011-07-03]. Dostupné z: <http://dev.w3.org/html5/markup/article.html#article>

[4] HICKSON, Ian. *Web Storage* [Online]. 7 July 2011 [cit. 2011-06-28]. Dostupné z: <http://dev.w3.org/html5/webstorage/>

[5] HICKSON, Ian. *The WebSocket API* [Online]. 7 July 2011 [cit. 2011-06-27]. Dostupné z: <http://dev.w3.org/html5/websockets>

[6] SALGUEIRO, Carlos Feyt. *jWebSocket - The Open Source Java WebSocket Server* [Online]. [cit. 2011-06-17]. Dostupné z: <http://jwebsocket.org/>

[7] MEYER, Eric. *CSS kompletní průvodce*. Brno : Zoner Press, 2007. ISBN 978-80-86815-64-0.

[8] BOS, Bert. *CSS current work & how to participate* [Online]. [cit. 2011-07-12]. Dostupné z: <http://www.w3.org/Style/CSS/current-work>

[9] *CSS Reference* [Online]. [cit. 2011-05-11]. Dostupné z: <http://www.w3.org/Style/CSS/current-work>

[10] HENDERSON, Cal. *PEAR::Flickr_API* [Online]. [cit. 2011-05-02]. Dostupné z: <http://code.iamcal.com/php/flickr/readme.htm>

[11] *CSS Compatibility and Internet Explorer* [Online]. [cit. 2011-07-20]. Dostupné z: <http://msdn.microsoft.com/en-us/library/cc351024(v=vs.85).aspx>

[12] *Create rounded corners using css easily….* [Online]. [cit. 2011-07-20]. Dostupné z: <http://erkcm.wordpress.com/2008/11/30/create-rounded-corners-using-css-easily/>

[13] *jQuery: The Write Less, Do More, JavaScript Library* [Online]. [cit. 2011-07-19]. Dostupné z: <http://jquery.com/>

[14] *Flickr Services* [Online]. 2011 [cit. 2011-05-11]. Dostupné z: <http://www.flickr.com/services/api/>

[15] *Chapter 10. HTTP Headers* [Online]. 2011 [cit. 2011-07-11]. Dostupné z: <http://trafficserver.apache.org/docs/v2/sdk/HTTPHeaders.html>

[16] *View HTTP Request and Response Header* [Online]. [cit. 2011-05-02]. Dostupné z: <http://web-sniffer.net/>

[17] *Taking Web Applications Offline with Gears* [Online]. [cit. 2011-07-21]. Dostupné z: <http://code.google.com/intl/cs-CZ/apis/gears/articles/take_app_offline.html>

[18] BOODMAN, Aaron. *Gears API Blog: Stopping the Gears* [Online]. [cit. 2011-07-18]. Dostupné z: <http://gearsblog.blogspot.com/2011/03/stopping-gears.html>

[19] *What's my IP?, What's my browser? - FindMeByIP - CSS3 & HTML5 Browser Support* [Online]. [cit. 2011-07-19]. Dostupné z: <http://fmbip.com/ >

[20] *Creative Commons — Attribution-NonCommercial-ShareAlike 3.0 Czech Republic — CC BY-NC-SA 3.0* [Online]. [cit. 2011-07-14]. Dostupné z: <http://creativecommons.org/licenses/by-nc-sa/3.0/cz/>

[21] PILGRIM, Mark. *Video - Dive Into HTML5* [Online]. 2011 [cit. 2011-07-16]. Dostupné z: <http://diveintohtml5.org/video.html>

[22] SCHENGILI-ROBERTS, Keith. *March 23rd SDIG Webinar: An Insight into the Benefits of SVG (Scalable Vector Graphics) DITA Writer* [Online]. [cit. 2011-06-28]. Dostupné z: <http://www.ditawriter.com/?p=239>

[23] *W3C HTML5 Logo* [Online]. November 1995 [cit. 2011-06-17]. Dostupné z: <http://www.w3.org/html/logo/>

B. Slovník zkratek

POJEM	VYSVĚTLENÍ
AJAX	Označení technologie webových aplikací, které mění obsah svých stránek bez nutnosti jejich znovunačítání (Asynchronous JavaScript and XML).
API	Zkratka anglického Application Programming Interface jež označuje rozhraní pro tvorbu aplikací.
CERN	Zkratka evropská mezinárodní organizace pro jaderný se sídlem v Ženevě.
cookies	Jedná se o HTTP cookes. Tato zkratka označuje malé množství dat, která WWW server pošle prohlížeči a ten je lokálně uloží. Při další návštěvě tohoto serveru prohlížeš pak tato data posílá zpět.
CSS3	Moderní vyvíjející se standard kaskádových stylů.
DOM	Z anglického Document Object Model – objektový model dokumentu.
ECMA	Zkratka mezinárodní organizace pro standardizaci (European Computer Manufacturers Association).
FLASH	Doplněk firmy Adobe, který umožňuje vložení interaktivních animací, prezentací na webu.
framework	Framework je softwarová struktura, která slouží jako podpora při programování a vývoji.
HTML	Zkratka HyperText Markup Language označující značkovací jazyk pro hypertext. Je jedním z jazyků pro vytváření a publikování stránek na internetu.
HTML5	Nové vyvíjející se standard HyperTextMarkupLanguage. Je ve strádiu návrhu organizací W3C.
HTTP	Zkratka "Hypertext Transfer Protocol". Jedná se o internetový protokol určený pro výměnu dokumentů ve formátu HTML.
IE9	Microsoft Internet Explorer 9
ISO	Zkratka mezinárodní organizace pro standardizaci (International Organization for Standardization).
MIME	Multipurpose Internet Mail Extensions byl původně navržen jako víceúčelová rozšíření internetové pošty, ale nyní jej používají i jiné internetové protokoly jako např. HTTP.

offline	Režim kdy je aplikace či celé zařízení odpojeno od internetu.
online	Režim kdy je aplikace či celé zařízení připojeno k internetu.
SGML	Zkratka Standard Generalized Markup Language znamenající standardní jazyk určený k formálnímu popisu struktury dokumentů.
SVG	Je zkratkou anglického Scalable Vector Graphics, česky škálovatelná vektorová grafika a umožňuje vekreslení obrazu vytvořeného z tvarů a křivek.
W3C	Zkratka (World Wide Web Consortium) mezinárodního konsorcia vyvíjející webové standardy.
WebGL	Grafická knihovna pro web.
WHATWG	Zkratka "Web Hypertext Application Technology Working Group" skupiny skládající se z firem jako Mozzila, Opera a Apple pracující na webových standardech.
XHTML	Zkratka anglického Extensible HyperText Markup Language. Značkovací jazyk jež měl vyhovovat podmínkám XML dokumentů při zachování zpětné kompatibility.
XMLHttpRequest	Jedná se o rozhraní umožňující webovým aplikacím (stránkám) komunikovat mezi částí klientskou a servrovou prostředníctvím HTTP protokolu. Nejčastěji využívají aplikace založené na AJAX technologii.

C. Přílohy

Příloha č.1 - Ukázková odpověď stromu XML_Tree

```
XML_Tree_Node Object (
       [attributes] => Array (
              [stat] => ok
       )
       [children] => Array (
              [0] => XML_Tree_Node Object (
                     [attributes] => Array (
                            [page] => 1
                            [pages] => 500
                            [perpage] => 2
                            [total] => 1000
                     )
                     [children] => Array (
                            [0] => XML_Tree_Node Object (
                                   [attributes] => Array (
                                          [id] => 5624589820
                                          [owner] => 59375315@N00
                                          [secret] => c6a6607c00
                                          [server] => 5023
                                          [farm] => 6
                                          [title] => IMG_0399.JPG
                                          [ispublic] => 1
                                          [isfriend] => 0
                                          [isfamily] => 0
                                   )
                                   [children] => Array ( )
                                   [content] => [name] => photo
                            )
                            [1] => XML_Tree_Node Object (
                                   [attributes] => Array (
                                          [id] => 5624589832
                                          [owner] => 45090623@N03
                                          [secret] => f9fb55941e
                                          [server] => 5230
                                          [farm] => 6
                                          [title] => 1999
                                          [ispublic] => 1
                                          [isfriend] => 0
                                          [isfamily] => 0
                                   )
                                   [children] => Array ( )
                                   [content] => [name] => photo
                            )
                     )
                     [content] => [name] => photos
              )
       ) [content] => [name] => rsp
)
```

Příloha č.2 - Instrukce k instalaci

Celá webová aplikace se nachází na přiloženém datovém médiu (obsah média v příloze č. 3) v ZIP archívu `code_xsvihe02.zip`. Její instalace se provádí nakopírováním obsahu archívu přes FTP či jiný přenosový kanál na webový prostor hostujícího serveru. Poté je potřeba případné změny klíče *Flickr API* v souboru `core.php`. Aplikace byla testována a vyžaduje hosting server typu *Apache*[29] a PHP verze 5.2.15 s následujícími vlastnostmi (Tabulka C.1) a tedy by bylo vhodné pro bezproblémový chod nastavit tyto vlastnosti také.

vlastnost	nastavení
expose_php	off
safe_mode	off
register_globals	off
magic_quotes_gpc	on
session.auto_start	off
knihovny pro přístup k MySQL	mysql, mysqli, pdo_mysql
error_reporting	E_ALL & ~E_NOTICE
display_errors	on
enable_dl	off

Tabulka C.1 – Nastavení vlastností *Apache* serveru kde byla webová aplikace testována.

Příloha č.3 – Obsah přiloženého média

Popis jednotlivých souborů a složek pro lepší přehled obsahu.

xsvihe02_code.zip ..... kód webové aplikace v archívu typu ZIP
- Flickr <složka> ..... *flickr API soubory*
- HTTP <složka> ..... *flickr API* soubory
- Net <složka> ..... *flickr API* soubory
- XML <složka> ..... flickr API soubory
- canvas <složka> ..... složka se soubory pro úpravu obrázku plátna canvas
  - img <složka> ..... složka s ikonami nástrojů
  - temp <složka> ..... složka s uloženými *seasions* a dočasně uloženými obrázky
  - draw.js ..... JavaScript soubor obsahující kreslící funkce - určený pro vložení
  - index.php ..... hlavní index soubor
  - init.js ..... soubor obsahuje inicializace metody pro práci s tlačítkovou lištou

img <složka> ..... obsahuje ikony použity na hlavní stránce

test <složka> ..... složka obsahující soubory využívané při testování prohlížeče

img <složka> ..... složka obsahující obrázky

index.php ..... hlavní soubor testování prohlížeče

modernizr-1.7.min.js ..... soubor obsahující testovací metody HTML5 a CSS3 podpory

style.css ..... vlastní CSS stylování

.htaccess ..... soubor nastavující parametry serveru Apache

Michroma-webfont.eot ..... soubor - font pro import přes CSS3

Michroma-webfont.svg ..... soubor - font pro import přes CSS3

Michroma-webfont.ttf ..... soubor - font pro import přes CSS3

Michroma-webfont.woff ..... soubor - font pro import přes CSS3

PEAR.php ..... *flickr API* soubor

about.html ..... soubor obsahující informace o aplikaci

cache.manifest ..... soubor nastavující chování souboru v *offline* režimu

core.php ..... jádro generování obrázků s parsováním XML odpovědí z *Flickr* databáze

img_logo.gif ..... obrázek loga

index.php ..... hlavní stránka

know_your_product-webfont.eot ..... soubor - font pro import přes CSS3

know_your_product-webfont.svg ..... soubor - font pro import přes CSS3

know_your_product-webfont.ttf ..... soubor - font pro import přes CSS3

know_your_product-webfont.woff ..... soubor - font pro import přes CSS3

master.php ..... hlavní stránka – bez *manifest* záznamu

menu.php ..... soubor generující menu – horní tlačítkovou lištu

offline.html ..... soubor aktivující se v režimu offline

style.css ..... soubor kaskádových stylů

viewer.js ..... obsahuje JavaScript funkce pomáhající zobrazení obrázků na hlavní obrazovce

xsvihe02_BP.pdf ..... elektronická podoba bakalářské práce