Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Vývoj aplikací s využitím HTML5

Jan Ivon

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Katedra informačního inženýrství

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Jan Ivon

Informatika

Název práce

Vývoj aplikací s využitím HTML5

Název anglicky

Application development using HTML5

Cíle práce

Cílem bakalářské práce je popsat možnosti vývoje aplikací s využitím HTML ve verzi 5 a souvisejících technologií. Dílčím cílem je vytvoření ukázkové aplikace demonstrující tyto možnosti.

Metodika

Metodika řešené práce je založena na studiu a analýze odborných informačních zdrojů. Na základě syntézy zjištěných poznatků budou popsány možnosti vývoje aplikací s využitím HTML5 a souvisejících technologií. Dále bude navržena a implementována ukázková aplikace, která bude využívat možností HTML5 a demonstrovat tak zjištěné poznatky.

Doporučený rozsah práce

35-40 stran

Doporučené zdroje informací

FREEMAN, Adam. Metro revealed: building Windows 8 apps with HTML5 and JavaScript. New York: Distributed to the book trade worldwide by Springer Science Business Media, c2012, xii, 90 p. ISBN 978-143-0244-899.

KESSIN Zachary. Programming HTML5 Applications Building Powerful Cross-Platform Environments in JavaScript. Sebastopol: O'Reilly Media, 2011. ISBN 978-144-9322-724.

LUBBERS, Peter, Brian ALBERS, Frank SALIM a Tony PYE. Pro HTML5 programming. 2nd ed. New York: Apress, c2011, xx, 332 p. Expert's voice in Web development. ISBN 9781430238645.

MUELLER, John. HTML5 programming with JavaScript for dummies. xiv, 394 pages. –For dummies. ISBN 9781118494189.

Předběžný termín obhajoby

2015/06 (červen)

Vedoucí práce

Ing. Jiří Brožek, Ph.D.

Elektronicky schváleno dne 10. 11. 2014

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 10. 11. 2014

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 10. 03. 2015

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Vývoj aplikací s využitím HTML5" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 15.3.2015 _____

Poděkování

Rád bych touto cestou poděkoval Ing. Jiřímu Brožkovi, Ph.D. za cenné rady a odborné konzultace při zpracování této práce.

Vývoj aplikací s využitím HTML5

Application development using HTML5

Souhrn

Tato bakalářská práce je tématicky zaměřena na problematiku vývoje aplikací s využitím HTML5. V teoretické části se práce zabývá popisem nejdůležitějších technologií, které se využívají při vývoji HTML5 aplikací. Praktická část práce se týká ukázkové aplikace, která slouží k demonstraci poznatků zmíněných v teoretické části. Vlastní využití technologií při tvorbě ukázkové aplikace je v praktické části následně popsáno.

Summary

This bachelor thesis is thematically focused on problems of application development using HTML5. The theoretical part of thesis deals with description of most importatnt technologies, which are used in development of HTML5 applications. The practical part of thesis is connected with sample application, which shows usage of knowledge from theoretical part. Usage of technologies in development of sample application is afterwards described.

Klíčová slova: html, html5, css, javascript, webové uložiště, file api, application cache, web workers

Keywords: html, html5, css, javascript, web storage, file api, application cache, web workers

Obsah

1 Úvod

HTML5 dělá Web prvotřídním prostředím pro vytváření reálných aplikací. Doplňuje existující JavaScriptové nástroje dalšími klíčovými rozšířeními pro API, které usnadňují vytváření aplikací, které působí samostatně, ne jako pouhé rozhraní pro operace se vzdáleným serverem. [1]

Web začal jako způsob sdílení souborů, uložených na webovém serveru, které se měnily pouze příležitostně. Vývojáři rychle přišli na to, jak tyto soubory generovat za chodu, což byl první velký krok směrem k budování aplikací. Dalším velkým krokem bylo přidání interaktivity do webového prohlížeče. JavaScript a DOM umožňoval vývojářům vytvářet dynamické webové stránky pomocí Dynamického HTML, od toho se však později opět upustilo. Dalším krokem byl Ajax, který opět poskytoval možnosti podobné jako Dynamické HTML a navíc umožňoval v malém množství komunikaci se serverem. [1]

HTML5 staví na 20 letém vývoji a zaplňuje některé kritické mezery. Na první pohled mnoho HTML5 změn přidává podporu pro služby, které dříve vyžadovely externí rozšíření, pravdou je však, že poskytuje JavaScriptu nástroje potřebné k vytváření samostatných aplikací s využitím HTML pro strukturu, CSS pro vizuální stránku a JavaScript pro logiku a chování. [1]

2 Cíl práce a metodika

2.1 Cíl práce

Cílem bakalářské práce je popsat problematiku vývoje aplikací s využitím HTML ve verzi 5. Cílem teoretické části práce je popsat jednotlivé technologie používané při vývoji HTML5 aplikací. V praktické části bude vytvořena a popsána aplikace demonstrující praktické využití popisovaných technologií z teoretické části práce.

2.2 Metodika

Metodika řešené práce je založena na studiu a analýze odborných informačních zdrojů. Z důvodů nedostatečných zdrojů k tomuto tématu v českém jazyce budou primárně využívány zdroje cizojazyčné. Na základě zjištěných zdrojů budou popsány jednotlivé technologie a možnosti jejich využití při vývoji aplikací v HTML5.

V praktické části práce bude využito zjištěných poznatků o technologiích využívaných k vývoji aplikací s využitím HTML5 k vytvoření ukázkové aplikace. Využití technologií při tvorbě ukázková aplikace bude následně v praktické části popsáno.

Práce by měla sloužit jako zdroj základních informací pro čtenáře, který se chce zabývat touto tématikou.

Texty podbarvené šedou barvou představují kód, jeho části nebo speciální názvy.

3 Teoretická východiska

3.1 HTML

HyperText Markup Language zkráceně HTML je značkovací jazyk určený k vytváření webových stránek, které jsou vzájemně propojeny sítí hypertextových odkazů a dohromady vytváří systém zvaný World Wide Web. HTML není jediný jazyk umožňující tvorvu webových stránek, rozhodně je však nejrozšířenějším. HTML stránky jsou obvykle rošířeny o použití CSS a JavaScript. Současnou verzí HTML je verze 5.

Každá HTML stránka nebo HTML dokument je prostý textový dokument, který se skládá ze série HTML elementů, které vytvářejí webový obsah a koncovky. Koncovka by standartně měla být `.html` nebo .htm není to však povinnost. HTML elementy jsou označovány pomocí tagů. [2]

HTML tagy jsou skryté příkazy uvnitř webové stránky, které rozhodují o tom, jak prohlížeč danou stránku zobrazí. Většina tagů musí mít dvě části, otevírací část a uzavírací část – jsou tzv. párové. Například element html, kde tag `<html>` je otevírací element a `</html>` je uzavírací element. Uzavírací element je stejný jako otevírácí, pouze navíc obsahuje dopředné lomítko a neobsahuje žádné atributy. Existuje však několik HTML tagů, které jsou nepárové a nepotřebují uzavírací část. Jedním z nepárových tagů je `<img>`. [3]

Některé HTML elementy mohou obsahovat i atributy, které slouží k poskytování dodatečných informací o elementu. Atributy jsou uváděny v otevírácím tagu daného elementu a mají formát klíč=hodnota. Většina atributů není povinná, existují však i výjimky, kde je atribut vyžadován nebo je doporučen z hlediska pravidel přístupnosti. Jeden z nejpoužívanějších elementů s povinným atributem je element obrázku, který požaduje povinný atribut `src`. Atribut `src` obsahuje cestu k souboru zobrazovaného obrázku. Přikladem atributu požadovaného z hlediska přístupnosti je atribut `alt`, který obsahuje alternativní text zobrazovaný v případě neúspěšného zobrazení obázku ze souboru. Stupeň přístupnosti dokumentu lze ověřit pomocí validátoru přístupnosti, který je dostupný na webu. [4]

```
<!DOCTYPE html>
<html>
   <head>
      <title>Titulek stránky</title>
    </head>
    <body>
        <h1>Nadpis</h1>
        <img src="cesta" alt="obrazek"></img>
    </body>
</html>
```

Kód 1 - Ukázka HTML struktury (zdroj: vlastní)

HTML kód by měl být vždy validní, tím se zaručuje, že bude kód správně interpretován v různých webových prohlížečích. Validita kódu také usnadňuje jeho revizi a je znakem každého profesionála. Validita HTML dokumentu se dá zkontrolovat pomocí webových validátorů, nejznámějším validátorem je validátor poskytovaný asociací W3C. [5]

3.1.1 HTML5

HTML5 je nástupcem HTML verze 4.01, která byla vydána v roce 1997, od té doby bylo HTML5 postupně představováno s oficiálním vydáním v roce 2014. Nová verze HTML byla budována tak, aby splňovala několik základních principů. HTML5 se snaží docílit menší závislosti na externích rozšířeních jako je Flash, příkladem toho je nový element pro video. Skriptování by mělo být nahrazeno, kdykoliv je to jen možné. Příkladem tohoto jsou například CSS transformace, které dovolují pomocí CSS vytvářet pokročilejší efekty než kdy dříve. Posledním principem je nezávislost na zobrazovacím zařízení. HTML5 se snaží o to, aby se obsah na všech zařízeních zobrazoval stejně. [6]

HTML5 přináší řadu nových tagů, kterými jsou například `<svg>` a `<canvas>` pro práci s grafikou nebo `<audio>` a `<video>` pro přehrávání audia a videa bez nutnosti externího rozšíření. HTML5 poskytuje také sadu API pro vytváření HTML5 aplikací, mezi nejzajímavější patří HTML Drag and Drop, HTML Local Storage nebo HTML Web Workers. Výraznou změnou je také zjednodušení HTML deklarace, která se nyní podstatně zkrátila. [7]

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

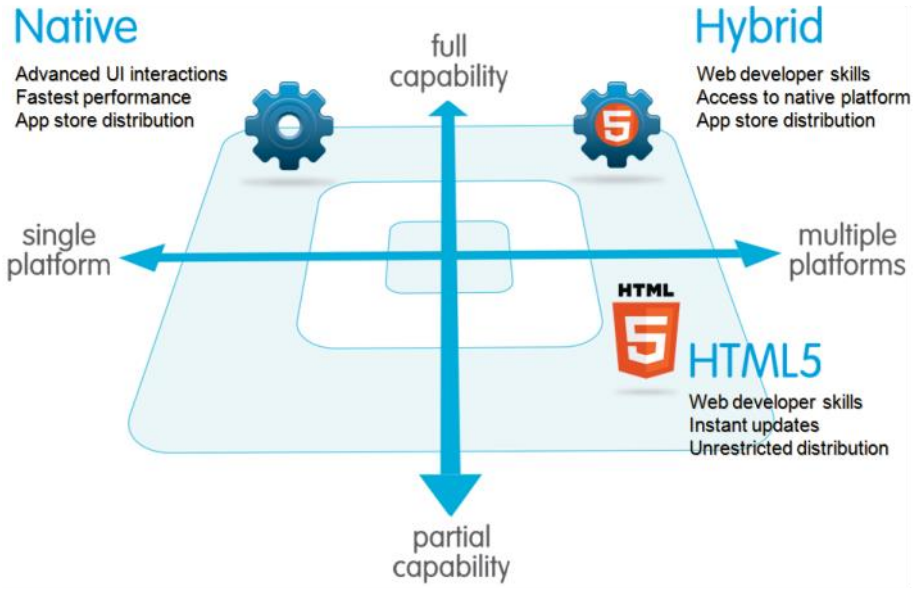
Kód 2 - HTML 4.01 doctype (zdroj: http://www.w3schools.com/tags/tag_DOCTYPE.asp)

```
<!DOCTYPE html>
```

Kód 3 - HTML5 doctype (zdroj: http://www.w3schools.com/tags/tag_DOCTYPE.asp)

HTML5 se dá kromě budování HTML5 aplikací využít i k budování hybridních mobilních aplikací nebo Metro aplikací pro Windows 8.

Hybridní mobilní aplikace umožňují využití předností HTML5 aplikací jako jsou přenositelnost a univerzálnost společně s nativním programováním, které přináší výhody přístupu k nástrojům dané platformy a optimalizaci výkonu. [8]



Obrázek 1 - Výhody platform (zdroj: https://developer.salesforce.com/page/Native,_HTML5,_or_Hybrid:_Understanding_Your_Mobile_Application_Development_Options)

Windows 8 Metro aplikace je speciální typ aplikace pro windows která se spouští v Metro režimu Windows. Tyto aplikace fungují stejně napříč platformou Windows, aplikace je spustitelná na klasickém osobním počítači, tabletu nebo chytrém telefonu. Jednou z možností vývoje těchto aplikací je využití HTML5, JavaScriptu a CSS. HTML5. [9]

3.2 CSS

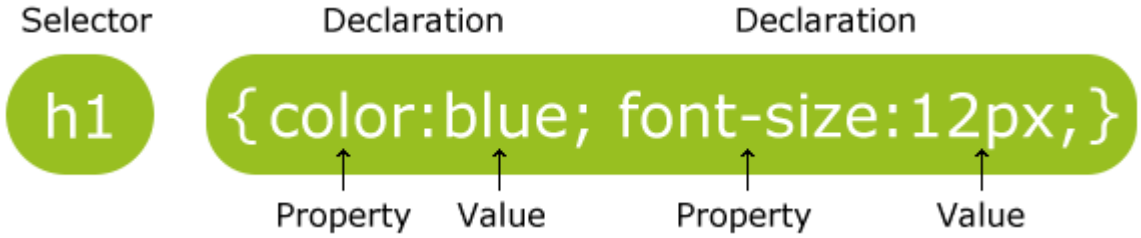
Cascading Style Sheets (kaskádové styly) jsou jazykem pro určení stylu zobrazení elementů HTML dokumentu. CSS vzniklo společně s verzí HTML 4.0, aby vyřešilo problém s tím, že HTML nikdy nemělo obsahovat tagy určující formátování. Toto umožňuje jednoduchou změnu designu celých stránek bez nutnosti zasahovat do samotného HTML, které obsahuje pouze strukturu formátovaného obsahu. Současnou verzí CSS je verze 3. [10]

CSS je obvykle uvedeno ve vlastním souboru, který je s HTML dokumentem spojen pomocí HTML tagu `<link>`, obsahujícího cestu k CSS souboru. Soubor CSS se běžně označuje koncovkou `.css`. CSS lze aplikovat i přímo v HTML dokumentu na jednotlivé elementy pomocí atributu `style`. [11]

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

Kód 4 - Externí deklarace CSS (zdroj: http://www.w3schools.com/css/css_howto.asp)

Stylování pomocí CSS probíhá tak, že je nejprve uvede selektor, který označuje co se bude stylovat. Selektorem může být například název elementu, třída nebo id. Následně se do složených závorek za selektor napíše požadový příkaz pro změnu stylu, přičemž do složených závorek lze psát více příkazů oddělených středníkem. [12]



Obrázek 2 - Struktura příkazu CSS (zdroj: http://www.w3schools.com/css/css_syntax.asp)

3.2.1 Selektory

Jak již bylo zmíněno, CSS stylování je aplikováno na obsah, který je vybrán pomocí selektorů, aby bylo stylování efektivní, je třeba se řídit pravidly pro používání selektorů. Selektorů existuje více druhů, každý druh má své specifika.

Selektor elementu

Tento selektor bude aplikovat změny na jeden nebo více HTML elementů se stejným názvem. [13]

```
ul {
    list-style: none;
    border: solid 1px #ccc;
}
```

Kód 5 - CSS selektor elementu (zdroj: http://www.sitepoint.com/web-foundations/css-selectors/)

Selektor ID

Selektor id je deklarován pomocí znaku `#`, který je následován řetězcem znaků označující název id. Řetězec znaků označující název id je definován vývojářem v HTML dokumentu v atributu `id` u požadovaného elementu. [13]

```
#container {
    width: 960px;
    margin: 0 auto;
}
```

Kód 6 - CSS id selektor (zdroj: http://www.sitepoint.com/web-foundations/css-selectors/)

```
<div id="container"></div>
```

Kód 7 - HTML atribut id (zdroj: http://www.sitepoint.com/web-foundations/css-selectors/)

Selektor třídy

Selektor třídy je deklarován podobně jako selektor id. Pro deklarování selektoru třídy se používá znak `.` následovaný řetězcem znaků označující název třídy. Řetězec znaků označující název třídy je definován vývojářem v HTML dokumentu v atributu `class` u jednoho nebo více požadovanch elementů. [13]

```
.box {
    padding: 20px;
    margin: 10px;
    width: 240px;
}
```

Kód 8 - CSS selektor třídy (zdroj: http://www.sitepoint.com/web-foundations/css-selectors/)

```
<div class="box"></div>
```

Kód 9 - HTML atribut class (zdroj: http://www.sitepoint.com/web-foundations/css-selectors/)

Sestupná selekce

Sestupná selekce dovoluje kombinaci dvou nebo více selektorů a díky tomu lze dosáhnout větší přesnosti u výběru stylovaného obsahu.

Selekce obsahuje dva nebo více selektorů oddělených mezerou. Stylovaný obsah je výběrem posledního uvedeného selektoru, který je přímým nebo nepřímým potomkem selektorů předchozích. [14]

```
#container .box {
   float: left;
   padding-bottom: 15px;
}
```

Kód 10 - CSS sestupná selekce (zdroj: http://www.sitepoint.com/web-foundations/css-selectors/)

```
<div id="container">
  <div class="box"></div> /* bude vybrán */

  <div class="box-2"></div>
</div>

<div class="box"></div>
```

Kód 11 - HTML potomci elementů (zdroj: http://www.sitepoint.com/web-foundations/css-selectors/)

Selekce potomků

Selekce potomků je stejná jako sestupná selekce, s tím rozdílem, že se selekce aplikuje pouze na přímé potomky. Tímto je dosahováno ještě vyšší přesnosti při výběru stylovaného obsahu. [13]

Selekce obsahuje dva nebo více selektorů oddělených znakem `>`. Stylovaný obsah je výběrem posledního uvedeného selektoru, který je přímým potomkem selektorů předchozích. [13]

```
#container > .box {
   float: left;
   padding-bottom: 15px;
}
```

Kód 12 - CSS selekce přímých potomků (zdroj: http://www.sitepoint.com/web-foundations/css-selectors/)

```
<div id="container">
  <div class="box"></div> /* bude vybrán */

  <div>
    <div class="box"></div>
  </div>
</div>
```

Kód 13 - HTML přímý potomci elementů (zdroj: http://www.sitepoint.com/web-foundations/css-selectors/)

Selektor atributů

Selekce se dá také provádět pomocí vybírání specifických atributů a jejich hodnot u elementů. Požadovaný atribut a jeho hodnota se uvádějí do hranatých závorek za název elementu. Lze vybírat také pouze atribut bez hodnoty. [13]

Následující příklad by aplikoval styl na všechny elementy typu `<input>` s atributem `type` a hodnout „text“.

```
input[type="text"] {
   background-color: #444;
   width: 200px;
}
```

Kód 14 - CSS selekce dle atributů (zdroj: http://www.sitepoint.com/web-foundations/css-selectors/)

Pseudo třídy

Pseudo třídy umožňují stylování elementu v momentě kdy je splněna určitá speciální podmínka. Podmínek existuje celá řada, mezi nejznámější patří `:hover`, která se aktivuje po najetí myší na element. [15]

```
a:hover {
    color: #FF00FF;
}
```

Kód 15 - CSS pseudo třída (zdroj: http://www.w3schools.com/css/css_pseudo_classes.asp)

3.3 JavaScript

Původně Internet umožňoval pouze statické webové stránky. Jediná dynamika, kterou statické stránky poskytují, jsou odkazy na stránky další. JavaScript byl původně zamýšlen jako nástroj pro docílení dynamiky webových stránek. Cílem bylo umožnit uživatelům interakci se stránkou, za což by uživatel zpětně obdržel data. Toto je i cílem dnešního

JavaScriptu, interakce se však rozrostla do takového měřítka, že se JavaScriptem dají místo obyčejných stránek vytvářet také webové aplikace. [16]

JavaScript je objektově orientovaný skriptovací jazyk většinou používaný na straně klienta, což znamená, že kód není potřeba kompilovat za účelem jeho vykonání. JavaScript místo toho vykonává řádek po řádku jednoduché příkazy. [6]

Omezení JavaScriptu vykonávání pouze jednoho příkazu v daném čase se dá částečně obejít pomocí technologie Web Workers, která je vysvětlena dále v dokumentu.

Umístit JavaScript do webové stránky nebo HTML5 aplikace je možné buď použitím tagu přímo v HTML dokumentu, nebo využitím externího souboru. První způsob využívá tagů `<script>` a `</script>`, mezi které se umístí požadovaný kód. Externí vložení skriptu se provádí pomocí stejných tagů, ale navíc se přidává atribut `src`, který obsahuje cestu k souboru JavaScriptu. Samotné tagy pro umístění JavaScriptu je možné vložit buď mezi tagy `<head>` a `</head>` nebo tagy `<body>` a `</body>`. Druhá varianta je lepší, protože se napřed načte a zobrazí webová stránka a až následně se vykonává skript. V opačném případě se napřed vykonává skript a to může výrazně zpomalit načtení zbytku HTML dokumentu. [7]

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
      <script src="myScript.js"></script>
    </body>
</html>
```

Kód 16 - Externí deklarace JavaScriptu (zdroj: http://www.w3schools.com/js/js_whereto.asp)

3.3.1 Podpora

Podpora JavaScriptu je zabudována do většiny dnešních prohlížečů včetně Internet Explorer, Mozilla Firefox nebo Google Chrome i prohlížečů na mobilních zařízeních. JavaScript lze však ručně vypnout, proto je vhodné uživatele upozornit, že má JavaScript vypnutý a web nebo aplikace nebude plně fungovat. [17]

```
<script type="text/javascript">
   document.write("Vítejte, JavaScript je dostupný.")
</script>
<noscript>JavaScript není zapnutý.</noscript>
```

Kód 17 - Kontrola podpory JavaScriptu (zdroj: https://css-tricks.com/snippets/javascript/detect-javascript-onoff-with-notification/)

3.4 HTML5 úložiště

HTML5 úložiště je specifikací pojmenováno jako Webové úložiště, avšak někteří výrobci prohlížečů tuto technologii označují jako Lokální úložiště nebo DOM úložiště. Tato technologie se soustřeďuje na poskytování trvalého lokálního úložiště pro webové aplikace, kterým tato možnost oproti aplikacím nativním až do příchodu HTML5 chyběla. [18]

Původně stránky zobrazené v prohlížeči neměly žádnou možnost, jak na uživatelském počítači uložit data. Způsobem nahrazení tohoto nedostatku mělo být využití cookies, to ale sebou nese několik nepříjemností. Zaprvé, cookie je přenášena při každém HTTP požadavku. To zbytečně navyšuje síťový přenos dat. Zadruhé, cookie data se posílají nezašifrovaná, což je jasné bezpečnostní riziko. Třetím a největším nedostatkem cookies je jejich velikost. Cookies jsou omezeny pouze na přibližně 4 kB dat. To je rozhodně nedostatek pro funkci úložiště, ale zároveň dost na zpomalení aplikace. [1]

HTML5 úložiště poskytuje trvalé lokální úložiště, ve kterém jsou data uchovávána ve formě datového typu string, v tomto formátu data přetrvávají i po zavření prohlížeče. Velikost úložiště neni přesně dána, protože závisí na jednotlivých prohlížečích. Úložiště je vždy sdíleno pro všechny stránky v jedné doméně. [1]

3.4.1 Podpora

Většina dnešních prohlížečů již HTML5 úložiště podporuje, v podpoře jsou však rozdíly, hlavně co se týká podporované velikosti úložiště. Důležité je vždy ověřit, zda webový prohlížeč HTML5 úložiště podporuje. Podpora se dá zjistit jednoduše pomocí následujícího kódu. [18]

```
if(typeof(Storage) !== "undefined") {
    // Webové uložiště je dostupné
} else {
    // Webové uložiště není dostupné
}
```

Kód 18 - Kontrola podpory HTML5 úložiště (zdroj: http://www.w3schools.com/html/html5_webstorage.asp)

4.0	8.0	3.5	4.0	11.5

Obrázek 3 - Podpora prohlížečů HTML5 úložiště (zdroj: http://www.w3schools.com/html/html5_webstorage.asp)

3.4.2 Objekty localStorage a sessionStorage

Moderní prohlížeče poskytují dva objekty pro úložiště, `localStorage` a `sessionStorage`. Každé může obsahovat data ve formě klíče a hodnoty. Mají stejné rozhraní a pracují totožně, s jedním hlavním rozdílem. Objekt `localStorage` je trvalý a restart prohlížeče s ním nic neudělá, zatímco `sessionStorage` objekt se resetuje, když je prohlížeč restartován. Kdy přesně se `sessionStorage` resetuje, závisí na daném prohlížeči. [1]

Data mohou být ukládána a opětovně získávána pomocí sady funkcí, které jsou pro `localStorage` i `sessionStorage` identické. Všechna data jsou ukládáná ve formě datového typu string.

localStorage

- Uložení - `localStorage.setItem`(klíč, hodnota);
- Opětovné získání – `localStorage.getItem`(klíč);
- Odstranění – `localStorage.removeItem`(hodnota);

```
if (localStorage.clickcount) {
    localStorage.clickcount =
Number(localStorage.clickcount) + 1;
} else {
    localStorage.clickcount = 1;
}
document.getElementById("result").innerHTML = "Stisknul
jste tlačítko " +
localStorage.clickcount + " krát.";
```

Kód 19 - Příklad práce s localStorage (zdroj: http://www.w3schools.com/html/html5_webstorage.asp)

sessionStorage

- Uložení - `sessionStorage.setItem`(klíč, hodnota);
- Opětovné získání – `sessionStorage.getItem`(klíč);
- Odstranění – `sessionStorage.removeItem`(hodnota);

```
if (sessionStorage.clickcount) {
    sessionStorage.clickcount =
Number(sessionStorage.clickcount) + 1;
} else {
    sessionStorage.clickcount = 1;
}
document.getElementById("result").innerHTML = "Stisknul
jste tlačítko " +
sessionStorage.clickcount + " krát za tuto seanci.";
```

Kód 20 - Příklad práce se sessionStorage (zdroj: http://www.w3schools.com/html/html5_webstorage.asp)

3.4.3 Storage události

HTML5 úložiště také obsahuje řadu událostí, které se aktivují v momentě volání funkce, která nějakým způsobem manipuluje s úložištěm, a to jak `localStorage`, tak i `sessionStorage`. Tyto eventy umožňují provést dynamické změny aplikace v závislosti na obsahu úložiště. [1]

3.5 Geolocation API

Geolocation API slouží k získání současné lokace z webového prohlížeče. Zeměpisná šířka a délka může být získána pomocí JavaScriptu, který může následně například zobrazit na mapě současnou polohu a zobrazit restaurace v blízkosti. Geolocation API dokáže sledovat více než jen polohu, například i rychlost, směr nebo nadmořskou výšku. [19]

3.5.1 Podpora

Před použitím Geolocation API je vhodné si zkontrolovat, jestli prohlížeč tuto možnost vůbec podporuje. Podpora se dá určit jednoduchou podmínkou. [20]

```
if (navigator.geolocation) {
  // Geolocation API je podporováno
}
```

Kód 21 - Kontrola podpory Geolocation API (zdroj: http://html5doctor.com/finding-your-position-with-geolocation/)

5.0	9.0	3.5	5.0	16.0

Obrázek 4 - Podpora prohlížečů Geolocation API (zdroj: http://www.w3schools.com/html/html5_geolocation.asp)

3.5.2 Získání současné polohy

Současná poloha se získá pomocí zavolání metody `getCurrentPosition`, to zahájí asynchroní požadavek detekce uživatelské polohy. Když je pozice zjištěna, tak se vykoná požadovaná callback funkce, která navrátí hodnotu. V případě chyby je možnost zadat další callback funkci jako druhý parametr. Metoda obsahuje ještě třetí parametr, který umožňuje nastavit přesnost, maximální dobu trvání zpracování požadavku nebo maximální stáří pozice z cache pozice. [21]

Vlastnost `PositionOptions.enableHighAccuracy` je boolean který indikuje, jestli se má aplikace snažit získat co nepjřesnější polohu. Toto může výrazně zpomalit odezvu, nejvíce u zařízení, které obsahují GPS. [22]

Vlastnost `PositionOptions.timeout` reprezentuje maximální délku času v milisekundách, po kterou je zařízení umožněno zpracovávat požadavek na získání polohy. [23]

Vlastnost `PositionOptions.maximumAge` indikuje maximální stáří v milisekundách, po které je možné použít pozici, která je již uložena v cache. [24]

```
navigator.geolocation.getCurrentPosition(function(positi
on) {
  do_something(position.coords.latitude,
position.coords.longitude);
});
```

Kód 22 - Příklad metody getCurrentPosition (zdroj: https://developer.mozilla.org/en-US/docs/Web/API/Geolocation/Using_geolocation)

3.5.3 Sledování současné polohy

Pokud je potřeba sledovat polohu průběžně, tak je vhodné použít metodu `watchPosition`, která volá callback funkci vícekrát a tím dovoluje webovému prohlížeči polohu obnovovat za pohybu. Metoda `watchPosition` má stejné parametry jako metoda `getCurrentPosition`. [21]

Metoda `watchPosition` navrací číslo id, které je jednoznačný identifikátor požadavku, pomocí tohoto čísla id a metody `clearWatch` se zastavuje požadavek na sledování. [21]

```
var watchID =
navigator.geolocation.watchPosition(function(position) {
  do_something(position.coords.latitude,
position.coords.longitude);
});
```

Kód 23 - Příklad metody watchPosition (zdroj: https://developer.mozilla.org/en-US/docs/Web/API/Geolocation/Using_geolocation#Getting_the_current_position)

```
navigator.geolocation.clearWatch(watchID);
```

Kód 24 - Příklad ukončení sledování metodou watchPosition (zdroj: https://developer.mozilla.org/en-US/docs/Web/API/Geolocation/Using_geolocation#Getting_the_current_position)

3.6 HTML Application Cache

HTML Application cache slouží k tomu, aby applikace nebo webová stránka byla dostupná i při výpadku internetu. Toto umožňuje soubor manifest, který obsahuje instrukce, kde má webová aplikace brát soubory. [25]

Webová aplikace se skládá ze souborů HTML, CSS a Javascriptu, které jsou mezi sebou provázány. Soubor manifestu obsahuje list URL adres s instrukcemi, které určují, jaké soubory si má aplikace u sebe udržovat lokálně. Soubor manifestu vždy kontroluje, jestli má aplikace uloženy nejnovější verze souborů, které má lokálně udržovat. Při nedostupnosti internetu manifest poslouží tomu, že začne odkazovat na lokální kopie souborů místo nedostupných souborů, které jsou uloženy na serveru. [1]

4.0	10.0	3.5	4.0	11.5

Obrázek 5 - Podpora prohlížečů Application Cache (zdroj: http://www.w3schools.com/html/html5_app_cache.asp)

3.6.1 Implementace manifestu

Aby aplikace mohla využívat manifestu, tak ho musí uvést v `<html>` elementu. Manifest obsahuje vlastní HTML atribut s názvem `manifest`, do kterého se vkládá cesta k souboru manifestu, běžně označovaného koncovkou `.appcache`.

```
<!DOCTYPE HTML>
<html manifest="demo.appcache">
    <body>
      Obsah dokumentu...
    </body>
</html>
```

Kód 25 - Příklad implementace manifestu (zdroj: http://www.w3schools.com/html/html5_app_cache.asp)

3.6.2 Struktura manifestu

Každý manifest začíná řádkou `CACHE MANIFEST`. Dále se pak manifest skládá ze tří základních částí, kterými jsou `CACHE`, `NETWORK` a `FALLBACK`.

CACHE

Soubory v této části manifestu budou co možná nejdříve staženy lokálně a budou spouštěny pouze lokálně. Tuto část lze psát i přímo za řádku `CACHE MANIFEST` bez použití hlavičky `CACHE`. [26]

NETWORK

Soubory uvedené zde mají být dostupné pouze online, proto nikdy za žádných okolností nebudou staženy a uloženy do lokální cache. Důvodem může být například skript na sledování návštěvnosti. Soubory lze jmenovat jednotlivě, ale většina aplikací si vystačí se symbolem `*` , který bude aplikovat toto pravidlo na všechny soubory. [26]

FALLBACK

Tato sekce slouží pro soubory, které se budou používat jako soubory náhradní za jiné, momentálně nedostupné. První část vždy označuje, co není dostupné a druhá, který soubor se načte místo původního. [26]

```
CACHE MANIFEST
# 2012-02-21 v1.0.0
/theme.css
/main.js
NETWORK:
login.asp
FALLBACK:
/html/ /offline.html
```

Kód 26 - Příklad manifestu (zdroj: http://www.w3schools.com/html/html5_app_cache.asp)

3.6.3 Aktualizace cache

Aktualizace cache probíhá ve dvou případech. Pokud uživatel vymaže soubory uložené v cache, v tom případě si je aplikace znovu stáhne podle manifestu. Další případ nastává ve chvíli, kdy je změněn samotný soubor manifestu. [27]

V případě, že se jedná o webovou stránku, která se může měnit, tak by mělo být u souborů uváděno číslo verze, aby bylo jednoznačně jasné, které soubory jsou novější. [27]

```
<script src="/script.js?svn_version=678"
type="text/javascript"></script>
```

Kód 27 - Uvádění verze souborů (zdroj: http://stackoverflow.com/questions/206783/when-does-browser-automatically-clear-javascript-cache)

3.6.4 Události

Ve chvíli, kdy se načte stránka s manifestem, prohlížeč spustí událost „checking“ na objektu `window.applicationCache`. Tato událost nezávisí na tom, jestli prohlížeč už stránku navšívil nebo ne. Následně, pokud je cache prázdná, spustí se „downloading“ událost, která se spouští i v případě že byl změněn manifest. V opačném případě, kdy manifest zmeněn nebyl, se spouští „noupdate“ událost. [1]

Celá událost aktualizace cache při stahování souboru je doprovázena řadou událostí, kterých lze využít ke spouštění vlastních scriptů, které mohou například zobrazovat průběh stahování a jeho současný stav. [27]

Checking

Událost informuje, že prohlížeč hledá aktualizace. Jedná se vždy o první událost v posloupnosti.

```
window.applicationCache.addEventListener('checking',
handleCacheEvent, false);
```

Kód 28 - Aktualizace cache - událost „checking" (zdroj: http://www.html5rocks.com/en/tutorials/appcache/beginner/)

Noupdate

Tato událost se spouští v případě, že se manifest nezměnil.

```
window.applicationCache.addEventListener('noupdate',
handleCacheEvent, false);
```

Kód 29 - Aktualizace cache - událost „noupdate" (zdroj: http://www.html5rocks.com/en/tutorials/appcache/beginner/)

Downloading
Byla nalezena aktualizace, prohlížeč začal stahování.

```
window.applicationCache.addEventListener('downloading',
handleCacheEvent, false);
```

Kód 30 - Aktualizace cache - událost „downloading" (zdroj: http://www.html5rocks.com/en/tutorials/appcache/beginner/)

Progress
Prohlížeč stáhnul data. Tato událost je spuštěna pro každý soubor, který je stažen.

```
window.applicationCache.addEventListener('progress',
handleCacheEvent, false);
```

Kód 31 - Aktualizace cache - událost „progress" (zdroj: http://www.html5rocks.com/en/tutorials/appcache/beginner/)

Cached
Všechny uvedené zdroje ze souboru manifestu jsou staženy.

```
window.applicationCache.addEventListener('cached',
handleCacheEvent, false);
```

Kód 32 - Aktualizace cache - událost „cached" (zdroj: http://www.html5rocks.com/en/tutorials/appcache/beginner/)

Updateready
Tato událost se spouští v případě, že soubory byly nově staženy.

```
window.applicationCache.addEventListener('updateready',
handleCacheEvent, false);
```

Kód 33 - Aktualizace cache - událost „updateready" (zdroj: http://www.html5rocks.com/en/tutorials/appcache/beginner/)

Obsolete
Soubor manifestu nebyl nalezen, následkem toho se automaticky maže cache.

```
window.applicationCache.addEventListener('obsolete',
handleCacheEvent, false);
```

Kód 34 - Aktualizace cache - událost "obsolete" (zdroj: http://www.html5rocks.com/en/tutorials/appcache/beginner/)

Error
Poslední událost v sekvenci, která nastává v případě, že se stala chyba.

```
window.applicationCache.addEventListener('error',
handleCacheError, false);
```

Kód 35 - Aktualizace cache - událost „error" (zdroj: http://www.html5rocks.com/en/tutorials/appcache/beginner/)

3.7 FILE API

Z bezpečnostních důvodů nebylo HTML dříve schopné pracovat se soubory na disku. Jedinou výjimkou byly HTML formuláře, které umožňovaly nahrát soubory na server. Jiné operace se soubory však možné nebyly. [1]

HTML5 nyní poskytuje možnost interakce s lokálními soubory, za použití nového API. Nové API je schopno například vytvořit náhled obrázku, zatímco se nahrává na server nebo umožnit aplikaci uložit referenci na soubor, zatímco uživatel není připojen k internetu. [28]

				
3.1	10.0	3.4	7.1	2.6

Kód 36 - Podpora prohlížečů File API (zdroj: vlastní)

3.7.1 Blob

Binnary large object, zkráceně Blob, je JavaScirptový objekt určený pro velké objemy dat. Tento objekt slouží k přesouvání velkých objemů dat. [29]

```
var aFileParts = ['<a id="a"><b id="b">hey!</b></a>'];
var oMyBlob = new Blob(aFileParts, {type :
'text/html'});
```

Kód 37 - Konstrukce Blob (zdroj: https://developer.mozilla.org/en-US/docs/Web/API/Blob)

3.7.2 Vkládání souborů

HTML5 poskytuje dvě základní možnosti vkládání souborů. Jednou je použití klasického formulářového elementu `<input>`, který je v HTML5 obohacen o nové hodnoty atributu `type`, včetně možnosti hodnoty `file`. Druhá možnost je novinkou v HTML5 a jmenuje se Drag and Drop.

Input

Pro vkládání pomocí elementu `input` stačí vytvořit formulář, který bude obsahovat již jmenovaný element s nastaveným atributem `type`, který bude mít hodnotu `file`. [30]

```
<input type="file">
```

Kód 38 - HTML input atribut "file" (zdroj: vlastní)

Drag and Drop

Dříve v HTML Drag and Drop nebylo možné a tato funkce vyžadovala využití softwaru Java. V HTML5 je to již minulostí a Drag and Drop lze zajistit za pomoci HTML, CSS a JavaScriptu.

Aby bylo možné vzít element a myší ho táhnout, musí element obsahovat atribut `draggable` s hodnotou `true`. Naštěstí většina dnešních prohlížečů automaticky tento atribut s hodnotou nastavuje bez toho, aby se o to musel programátor starat. [31]

```
<img draggable="true">
```

Kód 39 - HTML atribut "draggable" (zdroj: http://www.w3schools.com/html/html5_draganddrop.asp)

K docílení Drag and Drop efektu je potřeba využít HTML5 událostí, které umožňují použít JavaScript v požadovaný moment. HTML5 obsahuje pro Drag and Drop tyto události: [32]

`dragstart` – začátek táhnutí elementu myší

`drag` – táhnutí elementu myší

`dragenter` – táhnutý element vstupuje nad validním elementem pro vložení

`dragleave` – táhnutý element opouští validní element pro vložení

`dragover` – táhnutý element je táhnut přes validní element pro vložení

`drop` – táhnutý element je vložen do validního elementu pro vložení

`dragend` – konec táhnutí elementu myší

Pro dokončení efektu Drag and Drop se využívá objekt `DataTransfer` který obsahuje metody `setData` a `getData`. Tyto metody jsou použity pro uspěšný přenos taženého elementu nebo objektu do validního elementu. [33]

4.0	9.0	3.5	6.0	12.0

Obrázek 6 - Podpora prohlížečů Drag and Drop (zdroj: http://www.w3schools.com/html/html5_draganddrop.asp)

3.7.3 Práce se soubory

Jak již bylo zmíněno, dříve prohlížeče neumožňovaly JavaScriptu pracovat s HTML elementem `<input>`, který se používá k nahrávání souborů na server. Bylo to zapříčiněno obavou z toho, že by JavaScript mohl nahrát kterýkoliv soubor uložený na disku. Nové API však umožňuje JavaScriptu číst a pracovat se soubory, které uživatel do elementu `<input>` ručně vybral. [1]

Soubory nahrané pomocí elememtu `<input>` JavaScript vrací ve formě `FileList` objektu, který obsahuje seznam souborů ve formě `File` objektu. JavaScript nadále dokáže s `File` objektem pracovat a získávat z něj informace o názvu, velikosti, posledním datu úpravy a MIME typu. [1]

Soubor lze přečíst v celku, za použití `FileReader` objektu, nebo lze soubor rozdělit na menší části pomocí metody `slice` a poté číst soubor po částech z pole, které bylo metodou navráceno. [1]

Jelikož dnešní soubory mají běžně stovky MB, tak všechny metody objektu `FileReader` fungují asynchronně, aby nedošlo k dočasnému zamrznutí okna prohlížeče nebo aplikace. [1]

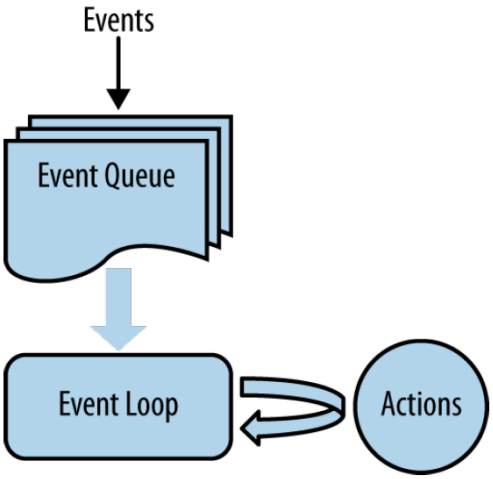
Objekt `FileReader` obsahuje čtyři metody. První metoda `ReadDataAsURL` upravuje soubor do podoby URL. Druhá metoda `ReadAsText` transformuje soubor na typ String s možnostní specifikovat formát, defaultně je ale použito kódování `UTF-8`. Metoda `ReadAsBinaryString` navrací data v binární podobě uložené v typu String. Poslední metoda `ReadAsArrayBuffer` navrací objekt `ArrayBuffer`. [1]

3.8 Web Workers

JavaScript je jazyk skriptovací, což znamená, že se vykonává řádek po řádku a vše funguje v jednom hlavním vlákně. Toto dříve nebyl žádný problém ale s růstem internetu a aplikací se na JavaScript kladly stále větší výpočetní nároky. [1]

Jako příklad nedostatku toho, že JavaScript pracuje pouze s jedním vláknem, se uvádí to, že moderní aplikace potřebují pracovat s několika prvky zároveň. Aplikace například potřebuje zpracovávat uživatelské rozhraní, dotazy a zpracovávat velké objemy dat. Bohužel kvůli omezení JavaScriptu totoho efektu nelze plně docílit a lze ho pouze částečně nahradit pomocí sady funkcí. [34]

Problém s jedním vláknem se dá částečně obejít použitím metod `setInterval`, `setTimeout()` nebo JavaScirptových událostí. Tyto způsoby ale stále běží pouze v jednom vlákně. To znamená, že se vykoná jedna část skriptu a až po dokončení se z fronty vybere další část, která se vykoná následně. HTML5 však přináší novinku, která dovoluje vícevláknové programování v JavaScriptu. [34]



Obrázek 7 - JavaScript smyčka událostí (zdroj: KESSIN Zachary. Programming HTML5 Applications Building Powerful Cross-Platform Environments in JavaScript)

Web Workers slouží k vytvoření skriptu, který dokáže běžet na pozadí aplikace, což dovoluje zpracovávat velké objemy dat bez přerušení odezvy uživatelského rozhraní a dalších prvků. Web Workers nepracují jako tradiční vlákna jako například v programovacím jazyku Java, kde si vlákna sdílejí stavy, místo toho si v JavaScriptu Web Workers posílají pouze zprávy. Díky tomuto je dosáhnuto parallelismu. [35]

Web Workers nemají stejné možnosti jako hlavní běžící skript. Web Workers nemohlou přistupovat k objektům `window`, `document` a `parent`. [35]

3.8.1 Podpora Web Workers

Jestli prohlížeč podporuje Web Workers lze zjistit následujícím skriptem.

```
if(typeof(Worker) !== "undefined") {
    // Web Workers mají podporu
} else {
    // Web Workers nemají podporu
}
```

Kód 40 - Kontrola podpory Web Workers (zdroj: http://www.w3schools.com/html/html5_webworkers.asp)

4.0	10.0	3.5	4.0	11.5

Obrázek 8 - Podpora prohlížečů Web Workers (zdroj: http://www.w3schools.com/html/html5_webworkers.asp)

3.8.2 Využití Web Workers

Grafika

HTML elementy `<svg>` a `<canvas>` umožňují manipulaci s grafikou ve vysokém rozlišení, operace s touto grafikou však bez Web Workers není možná, protože se jedná o operace, které zabírají čas a do menších úseků by se rozdělovaly těžko. S příchodem Web Workers je ale možné bez větších problémů dělat úpravy grafiky, které by dříve dočasně zastavily chod uživatelského rozhraní programu. [1]

Mapy

Další využití je možné v mapách, s příchodem HTML API Geolocation je nyní lehké si zjistit uživatelskou polohu. Se zjišťováním polohy lze například poskytovat instrukce cesty, nebo měřit kolik uživatel za den ujde. [1]

Aby aplikace mohla uživateli dávat přesně informace, tak je nutné, aby využívala Web Workers. Práci je potřeba rozdělit do několika prvků, přičemž se žádný nesmí zastavit. Uživatelské rozhraní musí neustále dodávat aktuální informace, kde se uživatel nachází. API pro zprocování map musí neustále sledovat pohyb uživatele, zatímco je potřeba neustále porovnávat aktuální pozici se stanovenou trasou. [1]

3.8.3 Práce s Web Workers

Vytvoření Web workeru

Web worker se vytváří pomocí konstruktoru `Worker`, který obsahuje cestu k souboru, kde je uložený skript požadovaného web workeru. Web Workers musejí být uloženy v samostatném JavaScriptovém souboru. Pokud soubor uvedený v konstruktoru existuje, tak se vytvoří nové vlákno pro web worker, které začne pracovat po stažení souboru. V případě, že soubor neexistuje, konstruktor vrátí chybu. [34]

```
if(typeof(w) == "undefined") {
    w = new Worker("demo_workers.js");
}
```

Kód 41 - Vytvoření web workeru (zdroj: http://www.w3schools.com/html/html5_webworkers.asp)

Komunikace s Web Workers

Komunikace mezi web workerem a jeho rodičem probíhá pomocí metody `postMessage`. Metoda je schopna předávat typ string, Boolean nebo číselné typy. [1]

Následující příklad ukazuje, jak jsou data předávána mezi hlavním skriptem a web workerem. Hlavní skript posílá data web workeru pomocí metody postMessage. Worker přijímá zprávu pomocí `onmessage` handleru nebo pomocí `addEventListener` na

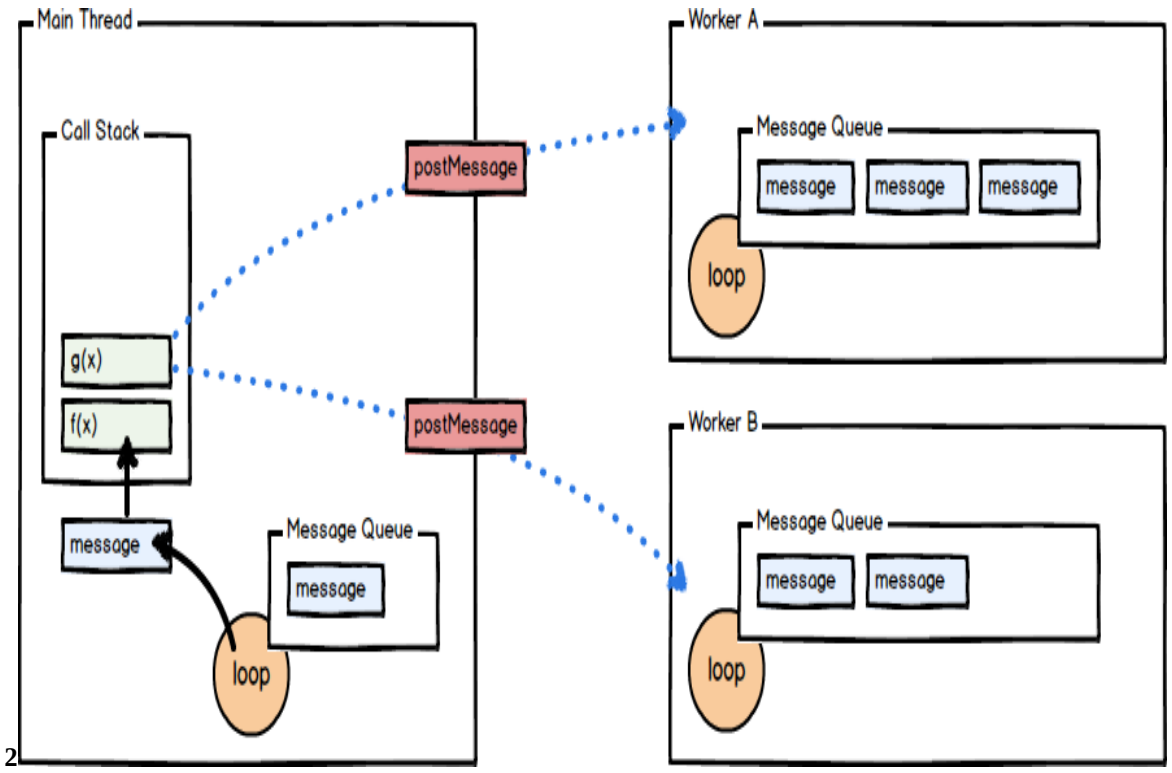
události `message`. A poté zpět posílá hlavnímu skriptu zprávu pomocí metody `postMessage`. Hlavní skript přijímá zprávu opět pomocí handleru jako to udělal web worker. Přijímání zpráv pomocí `addEventListener` je varianta doporučená odborníky. [34]

```
var worker = new Worker('doWork.js');

worker.addEventListener('message', function(e) {
  console.log('Worker said: ', e.data);
}, false);

worker.postMessage('Hello World');
```

Kód 42 - Web Workers komunikace - 1 (zdroj: http://www.html5rocks.com/en/tutorials/workers/basics/)

```
worker.addEventListener('message', function(e) {
  self.postMessage(e.data);
}, false);
```

Kód 43 - Web Workers komunikace - 2 (zdroj: http://www.html5rocks.com/en/tutorials/workers/basics/)



Kód 44 - Schéma komunikace Web Workers (zdroj: http://blog.carbonfive.com/2013/10/27/the-javascript-event-loop-explained/)

Nahrávání externích skriptů

Do Web Workers mohou být nahrávány externí skripty. Externí skripty se do Web Workers nahrávají pomocí metody `importScripts`, která má atribut, do kterého se vkládá string se jménem souboru nebo jeho cestou. Do jedné metody `importScripts` může být vložena i cesta k více skriptům najednou. [34]

```
importScripts('script1.js');
```

Kód 45 - Web Workers nahrání externího skriptu (zdroj: http://www.html5rocks.com/en/tutorials/workers/basics/)

Rušení web workeru

Od té doby, co je web worker vytvořen, neustále naslouchá zprávám od hlavního skriptu, aby se zbytečně neplýtvalo výpočetní silou počítače, tak je vhodné Web Workers rušit. Web worker se ruší pomocí metody `terminate`. [36]

```
webworker.terminate();
```

Kód 46 - Zrušení web workeru (zdroj: vlastní)

4 Vlastní práce

Tato kapitola popisuje postup, jak autor práce postupoval při tvorbě ukázkové aplikace, která je součástí bakalářské práce. Projekt ukazuje možnosti HTML5 technologií při tvorbě HTML5 aplikací. Technologie jsou popsané v teoretické části, ze které tato práce vychází.

Pro tvorbu ukázkové aplikace bakalářské práce bylo zvoleno vývojové prostředí NetBeans, které poskytuje přehledné prostředí pro tvorbu HTML5 aplikací. Aplikace byla testována na lokálním počítači v prostředí NetBeans, reálné využití by však bylo na webovém serveru. Pro testování aplikace je nutné využít vývojového prostřední NetBeans nebo Webového serveru. Veškerý kód aplikace je dostupný na CD, které je přiloženo k bakalářské práci.

4.1 Požadavky

Jak již bylo zmíněno, cílem je vytvoření ukázkové HTML5 aplikace, která bude demonstrovat zjištěné poznatky z teoretické části. Bude se jednat o aplikaci, do které je možné pouhým táhnutím myši vložit obrázek z plochy daného počítače. Při použití aplikace na mobilním telefonu se k vložení obrázku použije tlačítko. Na vložený obrázek bude možnost aplikovat sadu matematických filtrů a následné zobrazení upraveného obrázku místo obrázku původního. Aplikace filtrů bude probíhat za použití HTML5 technologie Web Workers, které umožní zpracování obrázku ve vlastním vlákně JavaScriptu. Délka doby aplikace matematického filtru na obrázek se bude zaznamenávat a výsledná doba trvání se uživateli zobrazí. Obrázek bude možno uložit do lokálního uložiště, po ukončení relace bude obrázek stále uložen v lokálním uložišti. Při další interakci s aplikací bude mít uživatel možnost obrázek z lokálního uložiště opěr nahrát a pokračovat v práci. Aplikace bude také využívat technologie Application Cache, která umožní ukládání souborů aplikace do lokální cache. Design aplikace bude vytvořen za účelem jednoduchosti, přehlednosti a responzivnosti dle typu obrazovky zařízení.

4.2 Popis struktury projektu

Projekt se skládá z několika základních souborů a také odkazuje na externí knihovny pomocí reference, všechny knihovny na které je odkazováno, jsou dostupné na Internetu. Nejdůležitější výchozí částí projektu je HTML dokument s názvem `index.html`, který obsahuje elementy uživatelského rozhraní. Styly pro elementy uživatelského rozhraní jsou uloženy v souboru `style.css`, na který je z HTML dokumentu odkazováno. Soubor `style.css` se nachází ve speciálním adresáři `css`, který se nachází v kořenovém adresáři uvnitř projektu. Kromě klasického CSS je

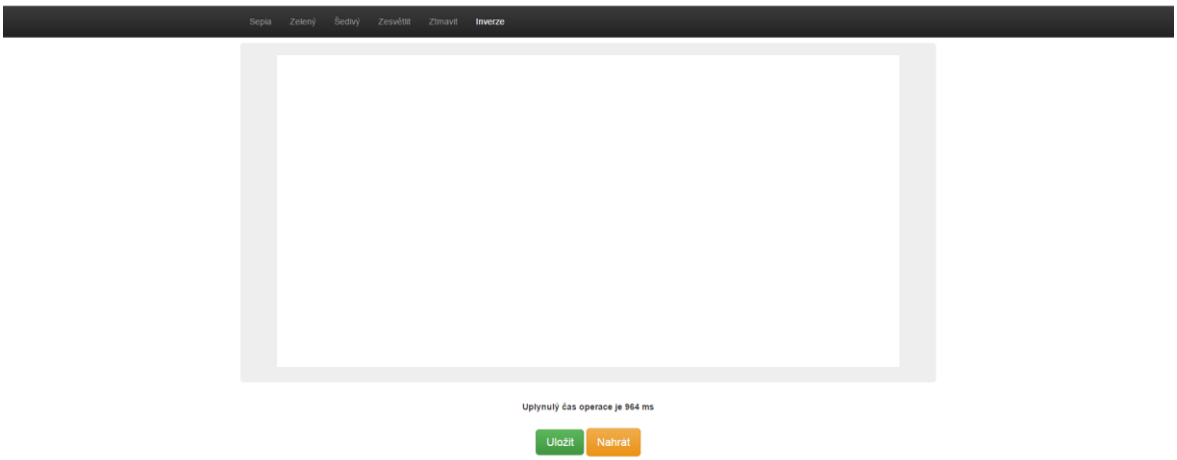
v projektu ještě využita knihovna Bootstrap, na kterou je taktéž odkazováno v HTML dokumentu, knihovna není součástí projektu a je na ní odkazováno externě. Aplikace využívá JavaScript, jehož hlavní část je uložena v souboru `main.js`, na který je odkazováno z HTML dokumentu. Kromě klasického JavaScriptu je ještě v malé míře využita knihovna JQuery, na kterou je taktéž odkazováno z `index.html`. Další částí projektu, které využívá JavaScript jsou soubory filtrů, které musejí být odděleně, neboť to vyžaduje technologie Web Workers. JavaScriptové soubory filtrů lze nalézt ve speciálním adresáři `filter` uvnitř kořenového adresáře projektu. Posledním souborem projektu je `manifest.appcache`, který obsahuje isntrukce pro technologii Application Cache, na soubor je odkazováno z tagu `<html>` v HTML dokumentu.

4.3 Postup vývoje

V této části práce je popsán postup vývoje HTML5 aplikace. Popsání postupu zahrnuje vysvětlení technologií a kódu, se kterým pracují.

4.3.1 Design

Pro vývoj designu se v projektu využívají soubory `index.html`, `style.css` a externí knihovna Bootstrap. Pří vývoji designu byl kladen cíl na jednoduchost, přehlednost a responzivitu dle velikosti obrazovky a typu zařízení.



Obrázek 9 - Náhled aplikace na klasickém monitoru (zdroj: vlastní)

Horní část HTML stránky je tvořena elementem `navigation bar`, který má tag HTML `<nav>`. Navigation bar v sobě obsahuje `<div>` s třídou `container`, tato třída je

vyžadována knihovnou Bootstrap k docílení responzivnosti. Tento `<div>` v sobě obsahuje další dva elementy typu `<div>`. První element `<div>` slouži k uskladnění tlačítka, které se zobrazuje pouze v případě, že je okno prohlížeče zmenšeno na mobilní zobrazení. Po stisknutí tlačítka se zobrazí možnosti filtrování. Tlačítko obsahuje třídy z knihovny Bootstrap, které slouží k responzivnosti, dále tlačitko obsahuje elementy `<span>`, které slouží pouze jako ikonky v daném tlačítku. Druhý element typu `<div>` slouží k uskladnění listu `<ul>`, který v sobě obsahuje elementy typu `<a>`, na které je navázaná JavaScriptová funkce `filtr` pro aplikaci filtrů. Funkce pro aplikaci filtrů je spouštěna po kliknutí na odkaz s názvem filtru, funkce se volá pomocí atributu `onclick`. Volaná funkce obsahuje jeden parametr, kterým je název požadovaného filtru, samotná funkce se zpracovává pomocí JavaScriptu v souboru `main.js`.

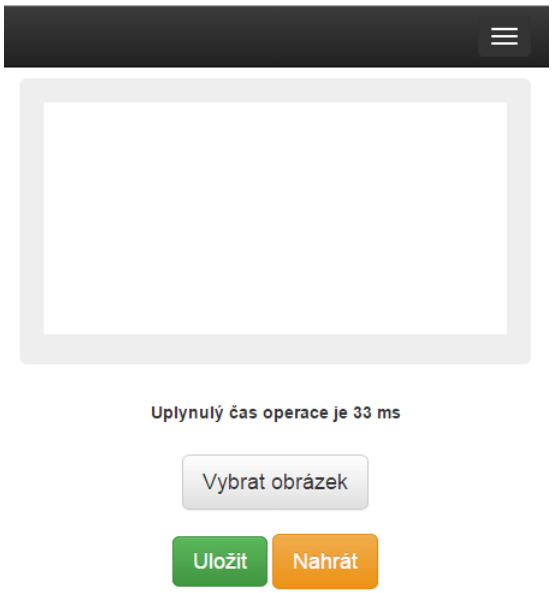
Část stránky se zobrazovací plochou pro obrázek je tvořena elementem `<div>`, který slouží jako `container` podobně jako v případě navigation baru. Tento element `<div>` v sobě obsahuje další element `<div>`, který náleží do třídy `jumbotron`, která slouží pro stylování pomocí Bootstrap i vlastního css. V obou těchto elementech je umístěn element `<canvas>`, který slouží jako vykreslovací plocha pro práci s obrázky.

Pod plochou pro vykreslování obrázků je umístěn `<div>` s třídou text-center, která slouží k umístění obsahu elementu na střed. Element `<div>` v sobě obsahuje element typu `<label>`, který je používám k zobrazení času operace s obrázkem.

Pod částí stránky se zobrazováním času operace je umísněn element `<div>` s třídou text-center, která slouží k umístění obsahu na střed stránky. Dále `<div>` obsahuje třídy `xs-col-12`, `sm-col-12`, `hidden-md` a `hidden-lg`, tyto třídy zajišťují zobrazení této části pouze na displejích tabletů a mobilních telefonů. Uvnitř elementu `<div>` je umístěn `<span>`, který slouží ke stylizaci elementu `<input>`, který je umístěn v elementu `<span>`. Element `<input>` slouží k vkládání obrázku do aplikace.

Ve spodní části stránky je umístěn `<div>` s třídou `text-center`, která slouží k umístění obsahu na střed stránky. Element `<div>` obsahuje dvě tlačítka, která slouží k nahrávání a ukládání obrázku z elementu `<canvas>` do webového uložiště. Tlačítka jsou elementy `<a>` stylované knihovnou Bootstrap. Tlačítko pro ukládání do webového úložiště je ve výchozím nastavení dočasně skryté, aby se zabránilo ukládání prázdného elementu `<canvas>`. Po stisknutí tlačítka pro uložení se volá funkce `saveCanvas`,

která zpracovává požadavek na uložení obrázku z elementu `<canvas>` do webového uložiště. Stisk tlačítka pro nahrávání spustí vykonávání funkce `loadCanvas`, která nahrává obrázek z webového uložiště zpět do elementu `<canvas>`.



Obrázek 10 - Náhled aplikace na obrazovce mobilního zařízení (zdroj: vlastní)

4.3.2 Drag and Drop

V aplikace je implementována funkcionalita Drag and Drop, která v této aplikaci dovoluje vkládat obrázky do elementu `<canvas>` pouhým přetažením myši obrázku z plochy počítače. Tato funkcionalita je částí HTML5 File API technologie, více o této technlogii se lze dozvědět z teoretická části bakalářské práce.

```
<body onload="prepare()">
```

Kód 47 - HTML onload funkce (zdroj: vlastní)

Po načtení elementu `<body>` v HTML dokumentu se volá funkce `prepare`, která má na starosti události, které probíhají během tažení obrázku myší a následné vložení obrázku do elementu `<canvas>`.

První událostí, kterou je potřeba ošetřit je událost `dragover` na elementu `<canvas>`. Při výchozím nastavení elementu `<canvas>` nelze použít přetažení obrázku na element, při přetažení by se objevila možnost obrázek zkopírovat do prohlížeče, tomu se musí zabránit metodou `preventDefault`, která je volána argumentem který jsme dostali z události.

```
canvas.on('dragover', function(e) {
    e.preventDefault();
});
```

Kód 48 - Událost „dragover“ (zdroj: vlastní)

Druhou událostí, kterou je potřeba ošetřit je událost `drop` na elementu `<canvas>`. Stejně jako v předchozí události je potřeba zabránit výchozímu chování prohlížeče metodou `preventDefault`. Dále se v této události pomocí `event.originalEvent.dataTransfer.files` získá obrázek, který byl přetažený do `<canvas>`, obrázek se ale nachází v poli, proto je nutné ho z pole vyjmout. Zároveň je potřeba podmínkou zkontrolovat, zda pole není prázdné.

Dalším krokem je vytvoření nového `FileReader` objektu, do kterého je pomocí metody `readAsDataURL` vložen obrázek, který jsme získali z pole. Tímto se aktivuje událost `reader.onload`, ve které vložíme do objektu `imageObj` pomocí `imageObj.src` data z objektu `FileReader` ve formě `base64` string, který je pro `imageObj` čitelný.

```
canvas.on("drop", function (e) {
    var files = e.originalEvent.dataTransfer.files;
    if (files.length > 0) {
        var file = files[0];
        if (typeof FileReader !== "undefined" &&
file.type.indexOf("image") != -1) {
            var reader = new FileReader();
            reader.onload = function (evt) {
                imageObj.src = evt.target.result;
            };
            reader.readAsDataURL(file);
            }
    }
    e.preventDefault();
});
```

Kód 49 - Událost „drop“ (zdroj: vlastní)

Načtením dat do `imageObj` se spouští funkce `drawImage`, která má na starosti vykreslení `imageObj` do elementu `<canvas>` pomocí metody `drawImage`.

V případě použití mobilního telefonu nebo tabletu je po vybrání obrázku tlačítkem spuštěna událost `change` a soubor je vybrán pomocí `event.target.files`, kód který se posléze vykonává je totožný s tím, který se vykonává při událost `drop`.

```
btnImage.on("change", function(e){
    var files = e.target.files;
});
```

Kód 50 - Událost „change“ (zdroj: vlastní)

4.3.3 Web Workers

V ukázkové aplikaci je implementována technologie Web Workers, tato technologie je použita ke zpracování filtrů aplikovaných na obrázky v elementu `<canvas>`. Díky této technologii je umožněno zpracovávat filtry obrázků v odděleném procesu, který nezastaví vykonávání hlavního skriptu. V případě že by filtry byly zpracovávány v hlavním vlákně, by existovala možnost dočasné nemožnosti užívat rozhraní aplikace. Více teorie o technologi Web Workers je v teoretické části bakalářské práce.

V případě, že uživatel klikne na jedno z tlačítek pro aplikaci filtru umístěných v horní části HTML dokumentu, spustí se funkce `filter`, zároveň se z tlačítka této funkci předá parametr, který filtr má být aplikován. Funkce `filter` je univerzální pro všechna tlačítka.

Po spuštění funkce `filter` se zkontroluje podpora pro technologii Web Workers, v případě že podpora chybí, tak konzole vypíše chybovou hlášku. Pokud je technologie Web Workers podporována, tak se vytvoří a spustí nový web worker, ve kterém se bude aplikovat filter na obrázek.

```
if(typeof(Worker) !== "undefined") {
    var worker = new Worker("filter/" + what + ".js");
}
```

Kód 51 - Vytvoření web workeru (zdroj: vlastní)

Do vytvořeného web workeru jsou poslána data o elementu `<canvas>` ve formě objektu a délka řetězce binárních dat elementu `<canvas>`, se kterým se ve filtru pracuje. Poslání dat do web workeru je vykonáno metodou `postMessage`.

```
var canvasData =
context.getImageData(0,0,canv.width,canv.height);
var len = canv.width * canv.height * 4;
worker.postMessage({ data: canvasData, length: len});
```

Kód 52 - Poslaní dat web workeru (zdroj: vlastní)

Data ve web workeru se přijímají pomocí události `onmessage`, přijatá data elementu `<canvas>` se převádí na binární řetězec. Následuje volání funkce `process`, která aplikuje filtr na binární data. Aplikace filtru probíhá pomocí cyklu, který prochází binární data a matematicky mění hodnoty jednotlivých pixelů. Po dokončení aplikace filtru se data posílají zpět do hlavního vlákna pomocí metody `postMessage`.

```
self.onmessage = function (e) {
    var canvasData = e.data.data;
    var binaryData = canvasData.data;
    var l = e.data.length;

    process(binaryData, l);
    self.postMessage({ result: canvasData});
};

var process = function (binaryData, l) {
    for (var i = 0; i < l; i += 4) {
        var r = binaryData[i];
        var g = binaryData[i + 1];
        var b = binaryData[i + 2];

binaryData[i] = (r*0.393)+(g*0.769)+(b*0.189);
binaryData[i + 1] = (r*0.349)+(g*0.686)+(b*0.168);
binaryData[i + 2] = (r*0.272)+(g*0.534)+(b*0.131);
}
```

Kód 53 - Skript web workeru (zdroj: vlastní)

V hlavním vláknu jsou data přijmuta pomocí události `onmessage` a jsou následně vložena do elementu `<canvas>` metodou `putImageData`. Po vložení dat do `<canvas>` je web worker ukončen metodou `terminate`.

```
worker.onmessage = function(e) {
    var canvasData = e.data.result;
    var context = canv.getContext("2d");
    context.putImageData(canvasData, 0, 0);
    worker.terminate();
};
```

Kód 54 - Navrácení hodnot z web worker (zdroj: vlastní)

V hlavním skriptu je během operace s filtrem ve web workeru měřen čas objektem `Date`. Poté, co je dokončena operace ve web workeru, je vytvořen nový objekt `Date`, na oba objekty se aplikuje metoda `getTime`, výsledný rozdíl mezi prvním a druhým objektem `Date` se zobrazí v uživatelském rozhraní jako čas potřebný ke zpracování.

```
var startTime = new Date();
startTime = startTime.getTime();

var finishTime = new Date();
finishTime = finishTime.getTime();
var timeDifference = finishTime - startTime;
```

Kód 55 - Měření času (zdroj: vlastní)

4.3.4 HTML5 Úložiště

HTML5 Úložiště v ukázkové aplikaci umožňuje ukládat element `<canvas>` a následně jej po ukončení aplikace a jejím opětovném spuštění opět zobrazit.

Ukládání

Po stisknutí tlačítka pro uložení se spouští funkce `saveCanvas`, tato funkce nejdříve zkontroluje podporu lokálního uložiště. V případě, že úložiště není podporováno, konsole zobrazí hlášku, v opačném případě se začne vykonávat skript pro ukládání.

Aby bylo možné uložit element `<canvas>`, je nutné získat data pomocí metody `toDataURL`. Po získání dat se využívá `localStorage` metody `setItem`, která ukládá data získaná z `<canvas>` do uložiště.

```
if(typeof(Storage) !== "undefined") {
    var dataURL = canv.toDataURL();
    localStorage.setItem("saved", dataURL);
}
```

Kód 56 - Uložení dat z canvas (zdroj: vlastní)

Nahrávání

Po stisknutí tlačítka pro nahrání obrázku se spouští funkce `loadCanvas`, tato funkce nejdříve zkontroluje podporu lokálního uložiště. V případě, že uložiště není podporováno, konzole zobrazí hlášku, v opačném případě se začne vykonávat skript pro nahrávání.

Prvním krokem je získání dat z lokálního uložiště pomocí `localStorage` metody `getItem`, zároveň s tím je vytvořen nový objekt `Image` v globální proměnné `imageObj`. Data získaná metodou `getItem` jsou nahrána do objektu `imageObj` pomocí `imageObj.src`.

Při nahrání dat do `imageObj` se spouští událost `imageObj.onload`, která vykresluje `imageObj` do elementu `<canvas>` metodou `drawImage`.

```
if(typeof(Storage) !== "undefined") {
    var dataURL = localStorage.getItem("saved");
    imageObj.onload = function() {
        context.drawImage(imageObj, 0, 0);
    };
    imageObj.src = dataURL;
}
```

Kód 57 - Nahrání dat do canvas (zdroj: vlastní)

4.3.5 Application Cache

V ukázkové aplikaci využívám technologie Application Cache, která umožňuje stahovat soubory webové aplikace do lokální cache, to umožňuje částečnou funkčnost aplikace i bez internetového připojení.

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

Kód 58 - Použití manifestu (zdroj: vlastní)

V HTML dokumentu je pomocí atributu `manifest` v elementu `<html>` nastavena cesta k souboru `manifest.appcache`. Soubor manifestu obsahuje instrukce, které soubory se mají stahovat lokálně a které se stahovat nemají.

```
CACHE MANIFEST
index.html
css/style.css
main.js

NETWORK:
*
```

Kód 59 - Manifest ukázkové aplikace (zdroj: vlastní)

V ukázkové aplikaci se pomocí manifestu ukládají do lokální cache počítače soubory `index.html`, `style.css` a `main.js`. Uložení těchto tří souborů do lokální cache umožní bez připojení k internetu aplikaci spustit a nahrát obrázky do elementu `<canvas>`, případně obrázky ukládat nebo nahrávat z lokálního uložiště.

Bez připojení k internetu nebude možné využívat filtry, jelikož z bezpečnostních důvodů nelze spouštět skripty Web Workers z lokálního počítače. Z toho důvodu nemá smysl skripty pro Web Workers ukládat do lokální cache.

5 Závěr a zhodnocení

Cílem této práce bylo seznámení s možnostmi vývoje aplikací za použití HTML5, toho bylo docíleno v kapitole 3. Nejdříve byl čtenář v kapitolách 3.1 až 3.3 seznámen se základy HTML, CSS a JavaScriptu. V podkapitolách teoretické práce 3.4 až 3.8 byly postupně popisovány možnosti HTML5. Podkapitola 3.4 popisuje ukládání souborů do lokálního počítače pomocí Webového úložiště. Následující podkapitola 3.5 zmiňuje možnosti sledování pomocí Geolocation API. Podkapitola 3.6 se soustředí na využívání cache lokálního počítače. V podkapitole 3.7 je popsána technologie File API a funkcionalita Drag and Drop. Poslední podkapitola 3.8 popisuje, jak pracovat s technologií Web Workers, která slouží ke spouštění nezávislých skriptů.

Dalším cílem práce bylo vytvoření a popsání praktického projektu, kterým byla ukázková aplikace sloužící k demonstraci zjištěných poznatků. Aplikace byla popisována v kapitole 4. Bylo uvedeno, v jakém softwarovém prostředí byla aplikace vyvíjena, jak se má aplikace testovat. Podkapitola 4.1 zmiňuje požadavky na funkčnost aplikace. Následující kapitola 4.2 se zabývá strukturou aplikace, popisuje jednotlivé soubory, ze kterých se aplikace skládá. Kapitola 4.3 popisuje vývoj ukázkové aplikace a využití technologií z teoretické části práce. Podkapitola 4.3.1 se zabývá popisem vytvořeného designu a popisem využitím HTML, CSS a knihovny Bootstrap. Podkapitola 4.3.2 popisuje využití File API k vytvoření funkcionality Drag and Drop. Následující podkapitola 4.3.3 popisuje implementaci technologie Web Workers. Podkapitola 4.3.4 se soustředí na implementaci technologie Webového úložiště. Poslední podkapitola 4.3.5 popisuje využití souboru manifest k lokálnímu cachování aplikace.

V budoucnosti by mohla být aplikace vylepšena přidáním větší škály filtrů nebo možností sdílet filtrované obrázky na sociální sítě. Dále by mohla přibýt funkce pro vrácení změn po aplikaci filtru.

6 Seznam použitých zdrojů

6.1 Monografické zdroje

[1] **KESSIN Zachary**. Programming HTML5 Applications Building Powerful Cross-Platform Environments in JavaScript. Sebastopol: O'Reilly Media, 2011. ISBN 978-144-9322-724.

[9] **FREEMAN, Adam.** Metro revealed: building Windows 8 apps with HTML5 and JavaScript. New York: Distributed to the book trade worldwide by Springer Science Business Media, c2012, xii, 90 p. ISBN 978-143-0244-899.

[10] **LUBBERS, Peter, Brian ALBERS, Frank SALIM a Tony PYE**. Pro HTML5 programming. 2nd ed. New York: Apress, c2011, xx, 332 p. Expert's voice in Web development. ISBN 9781430238645.

[16] **MUELLER, John**. HTML5 programming with JavaScript for dummies. xiv, 394 pages. --For dummies. ISBN 9781118494189.

6.2 Internetové zdroje

[2] HTML elements. w3schools. [Online] [Citace: 8. 2. 2015.] http://www.w3schools.com/html/html_elements.asp

[3] What is HTML. A Simple Guide to HTML. [Online] [Citace: 8. 2. 2015.] http://www.simplehtmlguide.com/whatishtml.php

[4] HTML attributes. w3schools. [Online] [Citace: 8. 2. 2015.] http://www.w3schools.com/html/html_attributes.asp

[5] Why validate. W3C. [Online] [Citace: 8. 2. 2015.] http://validator.w3.org/docs/why.html

[6] Scripting Language. techopedia. [Online] [Citace: 11. 2. 2015.] http://www.techopedia.com/definition/3873/scripting-language

[7] JavaScript Where To. w3schools. [Online] [Citace: 11. 2. 2015.] http://www.w3schools.com/js/js_whereto.asp

[8] Native, HTML5, or Hybrid: Understanding Your Mobile Application Development. Salesforce developers. [Online] [Citace: 9. 3. 2015.] https://developer.salesforce.com/page/Native,_HTML5,_or_Hybrid:_Understanding_Your_Mobile_Application_Development_Options

[11] HTM5 Introduction. w3schools. [Online] [Citace: 8. 2. 2015.] http://www.w3schools.com/html/html5_intro.asp

[12] CSS Introduction. w3schools. [Online] [Citace: 10. 2. 2015.] http://www.w3schools.com/css/css_intro.asp

[13] CSS Selectors. sitepoint. [Online] [Citace: 9. 3. 2015.] http://www.sitepoint.com/web-foundations/css-selectors/

[14] CSS/Selectors/combinators/descendant. W3C. [Online] [Citace: 9. 3. 2015.] http://www.w3.org/wiki/CSS/Selectors/combinators/descendant

[15] CSS Pseudo-classes. w3schools. [Online] [Citace: 9. 3. 2015.] http://www.w3schools.com/css/css_pseudo_classes.asp

[17] What is JavaScript. abouttech. [Online] [Citace: 11. 2. 2015.] http://javascript.about.com/od/reference/p/javascript.htm

[18] The Past, Present & Future of Local Storage For Web Applications. Dive Into HTML5. [Online] [Citace: 13. 2. 2015.] http://diveintohtml5.info/storage.html

[19] You are here (and so is everybody else). Dive Into HTML5. [Online] [Citace: 9. 3. 2015.] http://diveintohtml5.info/geolocation.html

[20] Finding your position with Geolocation. HTML5 doctor. [Online] [Citace: 9. 3. 2015.] http://html5doctor.com/finding-your-position-with-geolocation/

[21] Using Web Workers. Mozilla Developer Network. [Online] [Citace: 28. 2. 2015.] https://developer.mozilla.org/en-US/docs/Web/API/Geolocation/Using_geolocation

[22] PositionOptions.enableHighAccuracy. Mozilla Developer Network. [Online] [Citace: 28. 2. 2015.] https://developer.mozilla.org/en-US/docs/Web/API/PositionOptions/enableHighAccuracy

[23] PositionOptions.timeout. Mozilla Developer Network. [Online] [Citace: 28. 2. 2015.] https://developer.mozilla.org/en-US/docs/Web/API/PositionOptions/timeout

[24] PositionOptions.maximumAge. Mozilla Developer Network. [Online] [Citace: 28. 2. 2015.] https://developer.mozilla.org/en-US/docs/Web/API/PositionOptions/maximumAge

[25] HTML5 Application Cache. w3schools. [Online] [Citace: 14. 2. 2015.] http://www.w3schools.com/html/html5_app_cache.asp

[26] Let´s Také This Offline. Dive Into HTML5. [Online] [Citace: 16. 2. 2015.] http://diveintohtml5.info/offline.html

[27] **Bidelman, Eric.** A Beginner´s Guide to Using the Application Cache. HTML5 Rocks. [Online] [Citace: 16. 2. 2015.] http://www.html5rocks.com/en/tutorials/appcache/beginner/

[28] **Bidelman, Eric.** Reading files in JavaScript using the File APIs. HTML5 Rocks. [Online] [Citace: 17. 2. 2015.] http://www.html5rocks.com/en/tutorials/file/dndfiles/

[29] Blob. Mozilla Developer Network. [Online] [Citace: 17. 2. 2015.] https://developer.mozilla.org/en-US/docs/Web/API/Blob

[30] HTML <input> Tag. w3schools. [Online] [Citace: 19. 2. 2015.] http://www.w3schools.com/tags/tag_input.asp

[31] HTML5 Drag and Drop. w3schools. [Online] [Citace: 20. 2. 2015.] http://www.w3schools.com/html/html5_draganddrop.asp

[32] Drag and Drop. Mozilla Developer Network. [Online] [Citace: 20. 2. 2015.] https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Drag_and_drop

[33] **Bidelman, Eric.** Native HTML5 Drag and Drop. HTML5 Rocks. [Online] [Citace: 20. 2. 2015.] http://www.html5rocks.com/en/tutorials/dnd/basics/

[34] **Bidelman, Eric.** The Basics of Web Workers. HTML5 Rocks. [Online] [Citace: 25. 2. 2015.] http://www.html5rocks.com/en/tutorials/workers/basics/

[35] Using Web Workers. Mozilla Developer Network. [Online] [Citace: 28. 2. 2015.] https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API/basic_usage

[36] HTML5 Web Workers. w3schools. [Online] [Citace: 1. 3. 2015.] http://www.w3schools.com/html/html5_webworkers.asp

7 Přílohy

7.1 Obsah CD

Přílohové CD bakalářské práce obsahuje vypracovaný projekt bakalářské práce. Projekt se nachází v kořenovém adresáři přiloženého CD. Pro testování aplikace je nutné použít software NetBeans nebo Webový server.

7.2 Seznam použitých symbol a zkratek

HTML - HyperText Markup Language

CSS – Cascading Style Sheets

DOM – document object model

HTTP – Hypertext Transfer Protocol

URL – Uniform Resource Locator

API – Application Programming Interface

MIME – Multipurpose Internet Mail Extension

JSON – JavaScript Object Notation

7.3 Seznam obrázků

7.4 Seznam kódů