Masarykova Univerzita
Fakulta informatiky

Knižnica pre efektívne nahrávanie obrázkov na webový server

Bakalárska práca

Tomáš Bončo

Brno, Jar 2016

Namiesto tejto stránky vložte kópiu oficiálneho podpísaného zadania práce a prehlásenie autora školského diela.

Prehlásenie

Prehlasujem, že táto bakalárska práca je mojím pôvodným autorským dielom, ktoré som vypracoval samostatne. Všetky zdroje, pramene a literatúru, ktoré som pri vypracovaní používal alebo z nich čerpal, v práci riadne citujem s uvedením úplného odkazu na príslušný zdroj.

Tomáš Bončo

Vedúci práce: RNDr. Tomáš Obšívač

Poďakovanie

This is the acknowledgement for my thesis, which can span multiple paragraphs.

Zhrnutie

This is the abstract of my thesis, which can span multiple paragraphs.

Kľúčové slová

keyword1, keyword2, …

Obsah

Zoznam tabuliek

Zoznam obrázkov

1 Súčasné metódy nahrávania obrázkov

Aby sa dali veci robiť lepšie a efektívnejšie, je najprv potrebné preskúmať už vytvorené spôsoby a uvedomiť si ich nedostatky. Zamedzí sa tak opakovaniu chýb.

1.1 HTML tág <input>

Historicky prvým spôsobom je nepárový tág *<input type="file">*. Objavil sa už v roku 1997, keď konzorcium W3C vydalo štandard *HTML 3.2* [1]. Umožňuje nahrávanie súborov a teda aj obrázkov ako súčasť zaslaného formulára, a teda sa vždy musí vyskytovať uprostred tágu <form>. Scénar nahrávania prebieha nasledovne:

1. Užívateľ myšou klikne na tlačidlo "Zvoliť súbor"
2. Vyskočí modálne okno, z ktorého užívateľ vyberie obrázok
3. Užívateľ odošle formulár, obrázok je odoslaný na server, kde je spracovaný

Výhody tohto riešenia sú predovšetkým:

- Natívna podpora v prehliadači - nie sú potrebné žiadne ďalšie JavaScriptové súbory
- Jednoduché použitie

// TODO: schéma

1.2 FormData a XMLHttpRequest

XMLHttpRequest Level 2 pridáva nové rozhranie *FormData*, ktoré umožňuje vytvoriť reprezentáciu formulára v tvare kľúč-hodnota, čím umožňuje zaslať formulár pomocou *XMLHttpRequest* [2]. *XMLHttpRequest* je API, ktoré umožňuje prenášať dáta medzi prehliadačom a serverom bez toho aby nastal refresh [3].

Vďaka *FormData* je možné odstrániť limitáciu, kde *<input type="file">* musí byť vložený do formulára (<form>). Použitím *XMLHttpRequest*

eventu *onprogress* je možné sledovať proces nahrávania [4]. Scénar nahrávania môže vyzerať nasledovne:

1. Užívateľ myšou klikne na tlačidlo "Zvoliť súbor"
2. Vyskočí modálne okno, z ktorého užívateľ vyberie obrázok
3. Obrázok sa odošle na server, zatiaľ čo sleduje proces nahrávania. Keď je obrázok plne nahratý a spracovaný, odošle sa jeho odkaz naspäť užívateľovi do prehliadača.
4. Užívateľovi sa zobrazí nahraný obrázok, pokiaľ s ním nie je spokojný, postupuje od bodu 1, ak je spokojný odošle formulár. V tomto prípade nedochádza ku žiadnej manipulácií s obrázkom.

Výhody tohto riešenia sú predovšetkým:

- Nie je potrebné umiestnenie vo formulári
- Je možné zaslať obrázok na pozadí, čo umožňuje využitie v moderných webových aplikáciach, kde je refresh nežiadúci
- Počas nahrávania je možné zobraziť progres
- Po nahratí obrázka na server, je možné obrázok užívateľovi zobraziť. A teda ho vidí pred potvrdením formulára.

// TODO: schéma

1.3 Flash a Silverlight

// TODO: Netuším ako to vnútorne funguje, ale oba sú mŕtve. Zistiť.
Výhody:

- Je možné použiť dragdrop na zvolenie obrázka
- Je možné zobraziť obrázok pred odoslaním
- Počas nahrávania je možné zobraziť progress

2 Nevýhody súčasných metód

2.1 Podpora mobilných zariadení

V roku 2010, Steve Jobs, spoluzakladateľ a v tom čase aj výkonný riaditeľ Applu vydal vyhlásenie *Thoughts on Flash*[5], v ktorom vysvetľuje prečo zariadenia iPhone, iPad a iPad nepodporujú *Adobe Flash*. Kritizuje *Adobe Flash* z uzavrenosti, vysokej systémovej záťaže, slabej bezpečnosti a chýbajúcej podpore dotykových zariadení. BBC v roku 2012 informovala[6], že Adobe sťahuje *Adobe Flash Player* z *Google Play store*, čo dovtedy umožňovalo prehrávať *Adobe Flash* na mobilných zariadeniach. To znamená, že mobilné operačné systémy, ktoré dohromady zaberajú 96,7% trhu[7] nepodporujú *Adobe Flash*.

» Zdroj hovorí len o Smartphonoch, nie sú zahrnuté tablety. Použiť radšej: https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=8qpcustomd=1 » Nenašiel som žiaden dobrý zdroj, ktorý by priamo vravel, že Silverlight nie je podporovaný na iOS alebo Androide. Wikipedia rovno hovorí, že Silverlight je zastaraný. Opäť som ale nenašiel žiadne oficiálne tvrdenie.

2.2 Zbytočné súbory na serveri

Nahrávanie obrázkov bez toho, aby užívateľ najskôr obrázok videl, môže viesť k tomu, že užívateľ nahraje nesprávny obrázok. Tak isto môže vzniknúť problém, kde užívateľ kvôli nesprávnemu spracovaniu obrázku na serveri (napríklad nepsrávny orez viď. 2.3), môže zmeniť svoje rozhodnutie. Takto vznikajú na serveri súbory - obrázky, ktoré sa nikdy nepoužijú. Jedno z riešení tohto problému je pravidelné mazanie nevyužívaných obrázkov.

2.3 Orezanie obrázka

Pri spracovaní na serveri zvyčajne dochádza k zmenšeniu a orezaniu obrázka. Spravidla sa obrázky orezávajú na stred. To však môže byť nesprávne, ako ukazuje obrázok XX (TODO), kde orezanie na stred odreže osobu - najdôležitejšiu časť obrázka. Čiastočným riešením je

využitie detekcie tvárí. Pokročilé riešenia si vyžadujú pokročilú analýzu obrazu, kde sa analyzuje kontrast a hrany. Vďaka tomu je možné detekovať východ slnka, alebo budovu na obrázku.

original

2.4 Prenášanie zbytočných dát

// TODO

3 Vytvorenie knižnice na nahrávanie obrázkov

3.1 Ciele

Ciele, ktoré sme stanovili pri vytváraní knižnice majú za cieľ maximalizovať pohodlie užívateľa, minimalizovať zaťaženie pre tvorcov stránky, sprístupniť knižnicu všetkým. Preto sme stanovili, že knižnica by mala:

- byť nezávislá od knižníc/frameworkov tretích strán, napr. *jQuery*,
- podporovať nahrávanie či už zvolením obrázka cez modálne okno alebo aj technológiou *DragDrop*,
- byť jednuduchá na implementovanie
- byť dobre rozšíriteľná o nové funkcie
- podporovať nahrávanie bez refreshu
- podporovať mobilné zariadenia
- podporovať nahrávanie viacerých obrázkov
- byť dobre zdokumentovaná

3.2 Riešenie

3.2.1 Predstavenie

Najskôr sme plánovali využiť *jQuery*, ale kvôli splneniu prvého cieľa (byť nezávislí od knižníc/frameworkov tretích strán) sme sa rozhodli využiť *Custom Element*. Vytvorili sme jeden, ktorý slúži na úpravu obrázka, <*x-cupe*> a jeden na podporu multiuploadu - <*x-cupe-gallery*>. Oba vychádzajú z rovnakého HTML, pričom <*x-cupe-gallery*> interne využíva <*x-cupe*>, preto sa v tejto kapitole zameráme na element <*x-cupe*>. Nahrávanie viacerých obrázkov je bližšie popísané v kapitole 3.5.

Scénar nahrávania môže vyzerať takto:

1. Užívateľ buď pomocou DragDrop nahraje obrázok alebo klikne na element a následne si z modálneho okna zvolí obrázok
2. Obrázok sa zmenší, oreže a zobrazí
3. Pokiaľ užívateľ nie je spokojný s obrázokom, pokračuje prvým bodom
4. Pokiaľ užívateľ nie je spokojný s orezaním, držaním myšy posúva obrázok, dokým nie je spokojný
5. Keď je užívateľ spokojný s obrázkom aj jeho orezaním odošle formulár, a až vtedy sa zmenšaná a orezaná verzia nahraje na server

Hoc sa v scenári spomína klikanie, knižnica podporuje aj dotykové zariadenia a teda všetky úkony je možné vykonať dotykom. Interne obe elementy využívajú nasledujúce HTML:

```
<canvas></canvas>
<input type="file" style="display: none">
<input type="text" style="display: none">
```

Element <*canvas*> sa stará o zobrazenie obrázka, a na prácu s ním sme vytvorili pomocnú triedu *XCupeCanvasElement*, pomocou ktorej vieme rýchlo zmeniť rozmery. Keď užívateľ klikne na <*x-cupe*>, element vytvorí nový klik na <*input type="file"*>, čo spôsobí, že sa otvorí modálne okno a užívateľ si môže zvoliť obrázok. Pre zjednodušenie práce s ním sme vytvorili triedu *XCupeInputFileElement*. Do <*input type="text"*> sa vloží textová reprezentácia obrázka, čo umožňuje jeho odoslanie cez POST. Opäť, aj pre tento element sme vytvorili pomocnú triedu *XCupeInputFileElement*. Detailné popísanie, ako knižnica funguje vrátane využitia jednotlivých elementov je ďalej popísané v sekcií 3.4.

3.2.2 Inštalácia

Použitie *Custom Elementu* nám umožnuje skutočne jednoduchý spôsob inštalácie, kde sa najskôr načítajú zdrojové súbory knižnice:

<link rel=ïmport"href="dist/x-cupe.html»

Následne stačí už len na príslušnom mieste v kóde vytvoriť element:

<x-cupe></x-cupe>

V prípade nahrávania viacerých obrázkov:

<x-cupe-gallery></x-cupe-gallery>

// TODO: schéma prepojenia jednotlivých elementov a knižníc. Bolo by to vhodné?

3.3 Použité technológie, služby a postupy

3.3.1 Custom Element a ShadowDOM

3.3.2 Canvas

Canvas je nový tág *HTML5*, ktorý umožňuje pomocou JavaScriptu vykresľovať grafy, úpravovať fotografií, animovať a dokonca aj spracovať videa. Využíva ho aj *WebGL* na hardvérovo akcelerovanú 3D grafiku na webe[8].

V našej práci ho využívame na zobrazovanie obrázka. Samotná knižnica už obsahuje potrebné nástroje na vkladanie a zmenšovanie obrázka, čo nám uľahčuje prácu a zároveň nám <*canvas*> umožňuje prístup k jednotlivým pixelom, čo odcenia najm tvorcovia pokročilých rozšírení (napr. detekovať nevyužívanú plochu obrázka alebo prevod do čierno-bielej verzie). Keď užívateľ presúva obrázok, kvôli orezaniu, prekresľujeme daný výrez do <*canvas*> pomocou *requestAnimationFrame*[9], pre maximálnu plynulosť.

3.3.3 DragDrop

3.3.4 File Reader

File Reader umožňuje čítať obsah súborov vo webovom prostredí. Tie môžu byť načítané buď pomocou tágu <*input*> alebo pomocou *DragDrop* (viď. 3.3.3)[10][1].

1. Citovaný MDN udáva ešte aj HTMLCanvasElement.mozGetAsFile(), ktorý je neštandardizovaný a v dobe písanie tejto práce podporovaný iba prehliadačom Firefox

Použili sme ho práve pre prečítanie zvolených súborov, ktoré sme následne zobrazili v <*canvas*> (viď 3.3.2).

3.3.5 Promise

Promise je objekt určený pre asynchrónne výpočty a výpočty naplánované na neskôr (napríklad cez *setTimeout* alebo *setInterval*). Nevracia priamo metódu, ale vracia "prísľub", že niekedy v budúcnosti hodnota bude dostupná[11]. V silno asynchrónnej aplikácií zabraňuje tzv. "callback hell", kde množstvo callbackov bráni zrozumiteľnosti a ladeniu kódu.

Promise sme používame miesto predávania callbackov, predovšetkým pri čítaní užívateľovho obrázku.

3.3.6 TypeScript

TypeScript je typová nadstavba JavaScriptu, ktorá sa prekladá do JavaScriptu. Tým, že sa rýchlo vyvíja okrem typovosti prináša aj funkcie *ECMAScript 2015 (ES6)*, čo umožňuje progrmátorom používať funkcie, ktoré v čase písania tejto bakalárskej práce ešte nie sú implementované v prehliadačoch.

Naša knižnica využíva TypeScript pretože umožňuje rýchlejšie odhaľovať chyby, vedie k písaniu rozhraní, čo môže značne pomôcť tvorcom rozšírení a tiež kvôli podpore *ECMAScript 2015*. Navyše jeho vlastnosť, že sa prekladá do JavaScriptu nijak neobmedzuje tvorcov rozšírení alebo jeho celkové použitie.

3.3.7 Mocha

3.3.8 GitHub

Počas tvorby sme používali Git a výsledok našej práce sme uverejnili v službe GitHub - webovej službe na správu Git repozitárov. // TODO: pokračovať; prečo sme to spravili?

3.4 Postup nahrávania a nastavenia

V tejto sekcií sa budeme samotnnému procesu nahratia obrázku, od jeho zvolenia, úpravy po odoslanie na server.

3.4.1 Inicializácia knižnice

3.4.2 Zvolenie obrázka

Užívateľ môže obrázok zvoliť dvoma spôsobmi - výberom z modálneho okna alebo pomocou *DragDrop*. V prípade nahrávania obrázka cez modálne okno, užívateľ musí kliknúť na <*x-cupe*> element, do ktorého chce obrázok nahrať. Interne sa tento klik prevedie na <*input type="file"*> a to spôsobí otvorenie modálneho okna. Po tom, čo si užívateľ zvolí obrázok, je tento súbor pomocou *File Reader* (viď. 3.3.4) prečítaný a pripravený na ďalšie spracovanie. V prípade, že chceme tento spôsob nahrávania zakázať, stačí nastaviť atribút *allow-select* na hodnotu *false*.

V prípade použitia DragDrop a teda označenie obrázka, jeho presunutie ponad cieľový element a následne pustenie myši/oddialenie prsta od dotykovej plochy, je postup podobný. Element (<x-cupe>) vie pomocou zachytenia udalosti *drop* zachytiť presúvaný obrázok a dalej ho rovnako ako pri modálnom okne pomocou *File Reader* prečítať. Ak chceme tento spôsob zakázať, je potrebné nastaviť atribút *allow-drop* na hodnotu *false*.

3.4.3 Zmenšenie a orezanie

Po zvolení obrázka spravidla prebieha jeho zmenšenie a orezanie. Pre túto činnosť je potrebné vedieť rozmery nahrávaného obrázka a tiež rozmery výsledného obrázka, pričom sú implementované tieto scenáre:

Nahrávaný obrázok je väčší a mal by sa zmenšiť; orezávanie je zapnuté

> Očakávame, že práve tento scénar bude najvyužívanejší. V tomto scenári sa obrázok pomerne zmenší tak, aby pomerne kratšia strana nahrávaného obrázka bola zhodná s príšlusnou stranou finálneho obrázka. To je z toho dôvodu, aby užívateľ mohol hýbať obrázkom v ose, ktorá je pomerne dlhšia (a teda v tej, v ktorej má posúvanie zmysel). Pre lepšie porozumienie uvádzame aj obrázok. // TODO obrázok

Nahrávaný obrázok je väčší a mal by sa zmenšiť; orezávanie je vypnuté

V tomto prípade sa snažíme vložiť celý nahrávaný obrázok do finálneho obrázka. Dosiahneme to pomerným zmenšením obrázka tak, aby pomerne dlhšia strana bola zhodná s príslušnou stranou finálneho obrázka.

Nahrávaný obrázok je menší a mal by si zväčšiť; orezávanie je zapnuté

Cieľom našej knižnice je, aby spravca stránky vždy dostal z prehliadača taký obrázok, aký požaduje. Nemusí potom upravovať obrázok na serveri. To však znamená, že môže nastať prípad, že užívateľ zvolí menší nahrávaný obrázok, než je požadovaný finálny obrázok. V tom prípade prebieha spracovanie obrazu rovnako, ako v prvom scenári, len miesto zmenšovania sa obrázok zväčšuje.

Nahrávaný obrázok je menší a mal si zväčšiť; orezávanie je vypnuté

Postupujeme rovnako ako v prípade s väčším obrázkom. Miesto zmenšovania však obrázok zväčšujeme.

Finálny obrázok má len jeden fixný rozmer

Tento scenár je vhodný, ak vytvárame napr. obrázkove galérie, pričom chceme, aby obrázky mali pevnú šírku a ľubovoľne dlhú dĺžku. Ak chceme variabilnú šírku, je potrebné nastaviť atribút *width* elementu <*x-cupe*> na hodnotu *-1*. V prípade, že chceme variabilnú dĺžku, je potrebné obdobne nastaviť parameter *height* na hodnotu *-1*. V tomto scenári sa obrázok pomerne zmenší podľa fixnej príslušnej strany finálneho obrázka. Druhá strana finálneho obrázka sa nastaví pomerne zmenšej dĺžke finálneho obrázka. V tomto scenári nie je možné obrázok orezávať.

Finálny obrázok nemá fixný rozmer

Tento scénar je vhodný, ak chceme aby užívatelia mohli zaslať obrázok v ľubovoľných rozmeroch a zároveň ho pred odoslaním mohli vidieť. Finálny obrázok bude mať rozmery nahrávaného obrázka a ten teda nie je zmenšovaný ani orezávaný. Pre vytvorenie tejto situácie treba nastaviť aj atribút *width* aj atribút *height* na hodnotu *-1*.

Obrázok sa orezáva posúvaním obrázka (viď 3.4.4) a základným nastavením je orezanie na stred. To sa dá zmeniť atribútom *align*, pričom valídne hodnoty pozostávajú z kombinácie slov *left*, *right*, *top*, *bottom* a *center*, napríklad: <*x-cupe align="left bottom"*>. Pri zadaní iba jednej osi pre zarovnanie sa automaticky predpokladá, že druhá sa má zarovnať na stred. Orezanie sa dá úplne vypnúť a to nastavením atribútu *crop* na hodnotu *false*. Vnútorne knižnica pracuje tak, že sa textové hodnoty prerátajú na odsadenie obrázku zľava a zhora, ale obrázok sa nijako nemodifikuje. To nastáva až pri vykreslení.

Takisto sme si všimli, že ak zmenšíme veľmi veľký obrázok, jeho kvalita je nízka. Oproti pôvodnému obrázku je zdanlivo ostrejší a to je spôsobené nevyhovujúcim spôsobom prekresľovania obrázka, tzv. *downsampling*-om. Problém je, že prehliadač musí z plochy napríklad 4x4 pixely vytvoriť iba jeden pixel a aby to spravil najrychlejšie, jeden z nich vyberie, pričom sa neberie do úvahy kontext. Preto sme aplikovali jednoduchý algoritmus, ktorý obrázok zmenší postupne. Počet prekreslení je možné nastaviť atribútom *quality* a jeho predvolená hodnota je 3. Čím vyššia táto hodnota je, tým dlhšie trvá proces od zvolenia obrázka po jeho vykreslenie. Keď je táto hodnota príliš nízka (napríklad 1), tak je slabšia kvalita. Zmenšený obrázok sa uloží, aby pri jeho ďalšej manipulácií ho nebolo potrebné opäť zmenšovať.

3.4.4 Posúvanie obrázka

Pokiaľ je povolené orezanie obrázka, je možné toto orezanie meniť posúvaním obrázka. Aby sme odlíšili kliknutie, kedy má vyskočiť modálne okno s možnoťou výberu obrázka a samotné posúvanie, nastavili sme limit, počas ktorého musí užívateľ držať myš stlačenú na 100ms. Rovnaký limit platí aj pre dotykové zariadenia. Po spustení udalosti presúvania sa vyrátava súčasná pozícia myšy a porovnáva sa s pozíciou na ktorej začalo presúvanie. Rozdielom týchto hodnôt získavame informáciu o tom, ako máme posunúť obrázok. Tento rozdiel sa následne priráta k orezaniu obrázka. Nakoniec už zmenšený obrázok s novým nastavením orezania vykreslíme s využitím *requestAnimationFrame*. Posúvanie obrázka sa dá vypnúť nastavím atribútu $allow_movena\ hodnotu$false

//TODO: Prečo neprebieha ukladanie ak zvyčajne prebieha, ak prebieha?

3.4.5 Odoslanie obrázka

3.5 Nahrávanie viacerých obrázkov

Nahrávanie viacerých obrázkov je možné pomocou párového tágu <x-cupe-gallery>. *Ten interne využíva elementy* <x-cupe>, *ktoré vytvára vždy pri zvolení nového obrázka, či už pomocou modálneho okna alebo pomocou* DragDrop. *Element* <x-cupe-gallery> *podporuje všetky atribúty ako* <x-cupe>, *avšak pri ich zmene sa táto zmena pošle všetkým* x-cupe *elementom. Výnimkou sú atribúty* allow-drop *a* allow-select, *ktoré sa aplikujú len pre* <x-cupe-gallery> *a v* <x-cupe> *sú vypnuté.*

3.6 Rozšíriteľnosť

Nakoľko naša knižnica používa na spracovanie a zobrazovanie obrazu tág <canvas>, *využíva a aj poskytuje možnosť pracovať s obrazom na úrovni jednotlivých pixelov. To umožňuje vytváranie pokročilých rozšírení, ktoré napríklad dokážu zmeniť farebné otiene, využívať detekciu hrán, rozpoznávanie tvárí a podobne.*

3.6.1 Systém rozšírení

Predstavme si dve rozšírenia - "zoom"(rozšírenie umožňujúce zväčšovať a zmenšovať obraz) a "black&white"(rozšírenie, ktoré prevedie obraz do čierno-bielej škály). Chceli sme, aby tvorca stránky vedel stanoviť, ktoré <x-cupe> *elementy budú používať "black&white", a ktoré nie. Tiež sme chceli, aby vedel nastaviť rozšírenie "zoom"pre všetky* <x-cupe> *elementy.* ***Naším cieľom teda bolo, aby rozšírenie bolo možné aplikovať aj pre jednotlivé (<x-cupe>) elementy, ale aj univerzálne - pre všetky (<x-cupe>) elementy****. Naším ďalším cieľom bolo, aby* ***rozšírenia bolo možné kombinovať****, a teda aby tvorca stránky vedel nastaviť obe rozšírenia - "zoomäj "black&white"na rovnaký* <x-cupe> *element. Naším posledným cieľom bolo, aby boli rozšírenia nezávislé moduly - samostatné súbory, ktoré o sebe nevedia (resp. nemusia vedieť).*

Zhodnotili sme existujúce riešenia a rozhodli sme sa, že rozšírenia budú funkcie, ktoré budú upravovať metódy našej knižnice. Tým dosiahneme "cibuľový efekt".

3.6.2 Cibuľový efekt

Pre lepšie vysvetlenie uvádzame vzorku kódu so vzorového rozšírenia "zoom"(viď. 3.6.4), na ktorom následne vysvetlíme, "cibuľový efektä jeho výhody.

```
var self = this;
var originalRnDImg = controller.readAndDrawImage; // (A)
controller.readAndDrawImage = function() // (B)
{
    return originalRnDImg.apply( self, arguments ) // (C)
    .then( function()
    {
        // ... (D)
    }
}
```

(A) *Do premennej* originalRnDImg *uložíme pôvodnú metódu* readAndDrawImage XCupeController *triedy.*

(B) *Prepíšeme pôvodnú metódu* readAndDrawImage *novou funkciou, ktorá*

(C) *spustí pôvodnú metódu, a keď tá prebehne úspešne, tak*

(D) *vykoná doplňujúci kód.*

Je potrebné si uvedomiť, že metóda readAndDrawImage *ostáva zmenená a keď k nej bude pristupovať ďalšie rozšírenie (napr. "black&white"popísaný v 3.6.1), opäť pridá ďalšiu vrstvu. Následne keď bude metódu volať* XCupeController*, zavolá modifikáciu z "black&white", ktorá zavolá modifikáciu z rozšírenia "zoom", ktoré zavolá pôvodnú metódu.*

Tým sme dosiahli, že:

1. *Je možné aplikovať súčasne niekoľko rozšírení na jeden* <x-cupe> *element.*

2. *Jednotlivé rozšírenia o sebe nevedia. Všetky pristupujú (v prípade ukážky) k* XCupeController.

3.6.3 Aplikovanie rozšírenia

Rozšírenie je funkcia, pričom odporúčame, aby vždy dávala možnosť aplikovať rozšírenie zvlášť na jednotlivé prvky ale tiež aj na všetky prvky. Vo vzorovom rozšírení "zoom"(viď. 3.6.4) sme to dosiahli prvým parametrom, ktorý ak je prázdny tak sa aplikuje rozšírenie na triedu XCupeController. V prípade, že prázdny nie je, predpokladáme, že obsahuje odkaz na už vytvorený element a teda rozšírenie aplikujeme na ňom. Pre plné porozumenie uvádzame príklad:

```
// Aplikuje zoom pre~v etky XCupe elementy
makeCupeZoomable();

// Aplikuje zoom pre~vybran  XCupe element
var cupeElement = new HTMLXCupeElement();
makeCupeZoomable( cupeElement );
```

3.6.4 Vzorové rozšírenie

Praktická časť tejto práce, v zložke plugins *obsahuje vzorové rozšírenie* zoom. *Toto rozšírenie má za úlohu umožniť užívateľom zväčšovať a zmenšovať obrázok pred odoslaním.*

Keď užívateľ obrázok vloží, vyráta sa pomer medzi zobrazovacou plochou a skutočnou veľkosťou obrázka a vyráta sa maximálne oddialenie. Za najnižšie oddialenie (alebo maximálne priblíženie) je určená hodnota, kde jeden pixel nahrávaného obrázka zodpovedá jednému zobrazenému pixelu. Následné točenie stredným stlačídlom myši mení pomer, akým sa pôvodný obrázok zmenší predtým, než sa vykreslí.

3.7 Spracovanie obrázka na PHP serveri

3.8 Dokumentácia

4 Validácia riešenia

4.1 Podpora mobilných zariadení

4.2 Jednotkové testy

4.3 Porovnanie s ďalšími riešeniami

5 Záver

5.1 Odporúčané použitie

5.2 Nápady na vylepšenie

Literatúra

[1] *W3C, "HTML 3.2 Reference Specification."* `http://www.w3.org/TR/REC-html32`*, 1997. cit. 2014-11-15.*

[2] *Mozilla Developer Network, "FormData."* `https://developer.mozilla.org/en-US/docs/Web/API/FormData`*. cit. 2015-11-14.*

[3] *Mozilla Developer Network, "XMLHttpRequest."* `https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest`*. cit. 2015-11-14.*

[4] *Mozilla Developer Network, "progress."* `https://developer.mozilla.org/en-US/docs/Web/Events/progress`*. cit. 2015-11-14.*

[5] *S. Jobs, "Thoughts on Flash."* `http://www.apple.com/hotnews/thoughts-on-flash/`*, 2010. cit. 2014-11-15.*

[6] *BBC, "Adobe Flash Player exits Android Google Play store."* `http://www.bbc.com/news/technology-19267140`*, 2015. cit. 2014-11-15.*

[7] *IDC, "Smartphone OS Market Share, 2015 Q2."* `http://www.idc.com/prodserv/smartphone-os-market-share.jsp`*, 2015. cit. 2014-11-15.*

[8] *Mozilla Developer Network, "Canvas."* `https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API`*. cit. 2015-11-21.*

[9] *Mozilla Developer Network, "Window.requestanimationframe()."* `https://developer.mozilla.org/en-US/docs/Web/API/window/requestAnimationFrame`*. cit. 2015-11-21.*

[10] *Mozilla Developer Network, "Filereader."* `https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise`*. cit. 2015-11-21.*

[11] *Mozilla Developer Network, "Promise."* `https://developer.mozilla.org/en-US/docs/Web/API/FileReader`. *cit. 2015-11-21.*