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

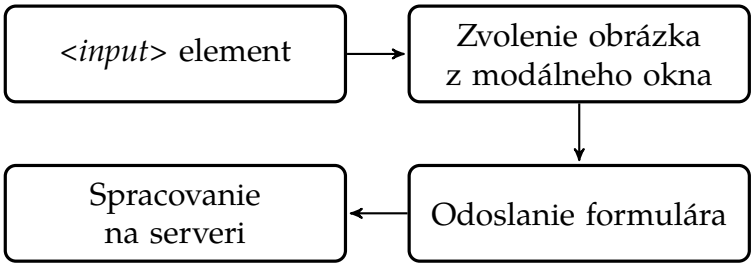
Zoznam obrázkov

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* [3]. Umožňuje nahrávanie súborov, a teda aj obrázkov, ako súčasť zaslaného formulára (musí sa vyskytovať uprostred tágu <form>). Scénar nahrávania prebieha nasledovne:



Obr. 1.1: Schéma nahrávania obrázka pomocou *<input>* elementu.

1. Užívateľ myšou klikne na *<input>* element.
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 prehliadačoch - nie sú potrebné žiadne ďalšie JavaScriptové súbory.
- Jednoduché na implementáciu.
- Podpora *Drag&Drop*.

1.2 FormData a XMLHttpRequest

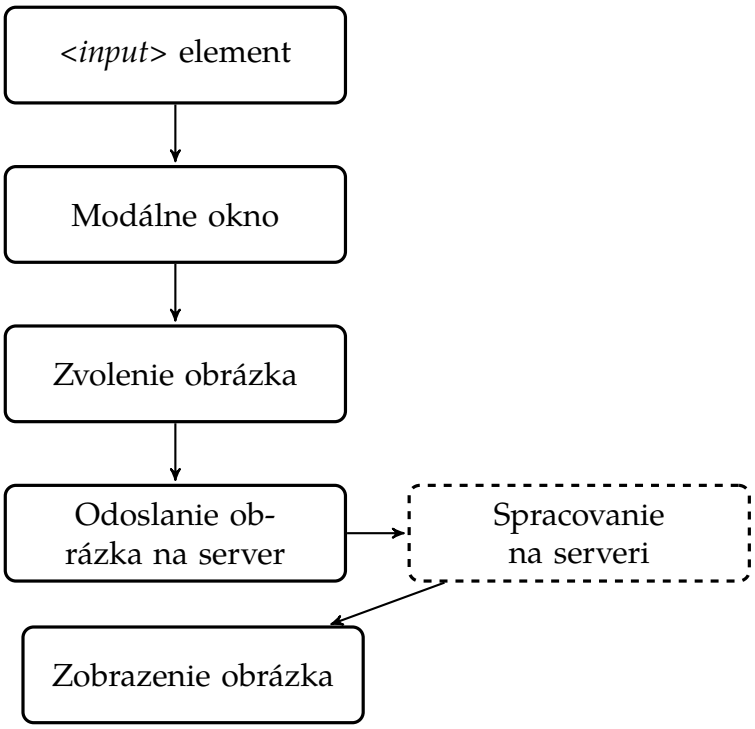
XMLHttpRequest Level 2 pridáva nové rozhranie *FormData*, ktoré umožňuje vytvoriť reprezentáciu formulára v tvare kľúč-hodnota, čím umožňuje zaslať formulár pomocou *XMLHttpRequest* [4]. *XMLHttpRequest* je API, ktoré umožňuje prenášať dáta medzi prehliadačom a serverom bez toho, aby nastalo obnovenie stránky [5].

Vďaka *FormData* je možné odstrániť obmedzenie, kde <*input type="file"*> musí byť vložený do formulára (<form>). Použitím *XMLHttpRequest* eventu *onprogress* je možné sledovať proces nahrávania [6]. Scénar nahrávania môže vyzerať nasledovne:

1. Užívateľ myšou klikne na <*input*> element.
2. Vyskočí modálne okno, z ktorého užívateľ vyberie obrázok.
3. Obrázok sa odošle na server, zatiaľ čo užívateľ sleduje proces nahrávania. Keď je obrázok plne nahraný 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 k žiadnej manipulácii 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 aplikáciách, kde je obnovenie stránky nežiaduce.
- Počas nahrávania je možné zobraziť proces v percentách.
- Po nahratí obrázka na server je možné obrázok užívateľovi zobraziť. Takto ho užívateľ vidí pred potvrdením formulára.
- Podpora *Drag&Drop*.



Obr. 1.2: Schéma nahrávania obrázka pomocou <*input*> elementu v kombinácií s *FormData* a XMLHttpRequest.

1.3 Riešenia založené na Adobe Flash a Microsoft Silverlight

V dobe písania tejto práce boli stále veľmi rozšírené riešenia, založené na technológii Adobe Flash (ďalej spomínaný už len ako Flash). Flash, pôvodne navrhnutý pre tvorbu multimediálneho obsahu - vektorovej grafiky, animácií, hier pre prehliadač - sa stal populárnym aj na tvorbu väčších aplikácií v prehliadači, pracovanie so vstupm z webovej kamery či mikrofónu, až po prehrávanie videí a zvuku.

O niečo podobné sa pokúšal Microsoft s technológiou Silverlight (ďalej len Silverlight). Obe technológie nepoužívajú Javascript, ale iné jazyky - v prípade technológie Flash ide o ActionScript a v prípade Silverlight ide o Visual Basic, C#, Ruby alebo Python. **V týchto technológiách by preto bolo možné napísať aj celú našu knižnicu**, avšak

nemohli by sme dosiahnuť podporu mobilných zariadení a obe technológie vyžadujú inštaláciu rozšírení tretích strán do prehliadača.

Hlavným dôvodom neúspechu technológie Flash a Silverlight sa stalo odmietnutie podpory od firmy Apple (pozri 2.1). V súčasnej dobe nie je podporovaný na mobilných zariadeniach s operačným systémom iOS, Android a Windows Phone. Samotné Adobe už prestáva ďalej Flash podporovať a zameriava sa radšej na HTML5[7].

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*[8], v ktorom vysvetľuje, prečo zariadenia iPhone, iPod a iPad nepodporujú *Adobe Flash*. Kritizuje *Adobe Flash* z uzavretosti, vysokej systémovej záťaže, slabej bezpečnosti a chýbajúcej podpore dotykových zariadení. BBC v roku 2012 informovala[9], 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[10] nepodporujú *Adobe Flash*.

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 obrázka na serveri (napríklad nesprávny orez, pozri 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 2.1, kde orezanie na stred odreže osobu - najdôležitejšiu časť obrázka. Čiastočným riešením je využitie detekcie tvárí. Pokročilé riešenia si však vyžadujú pokročilú analýzu obrazu, kde sa analyzuje kontrast a hrany. Vďaka tomu je možné detekovať východ slnka, alebo budovu na obrázku.

Obr. 2.1: Demonštrácia zlého automatického orezania obrázka.

2.4 Prenášanie zbytočných dát

Spracovanie obrázka obvykle prebieha na serveri. To však znamená, že ak je žiadaný len zmenšený a orezaný obrázok, tak väčšina dát, ktoré užívateľ preniesol, je zbytočná. Pokiaľ by zmenšenie a orezanie obrázka prebiehalo v prehliadači, tak sa prenesie menej dát, čo zrýchli nahrávanie obrázka na webový server a tiež ušetrí peniaze v prípade spoplatnených mobilných dát.

3 Validácia riešenia

3.1 Podpora mobilných zariadení

3.2 Porovnanie s ďalšími riešeniami

Vzhľadom k tomu, že knižníc na nahrávanie obrázkov je veľa, rozhodli sme sa porovnať tú našu s vybranými ostatnými. Porovnávali sme spôsob a rýchlosť nahrávania, spôsob spracovania a ponúkané možnosti. Všetky knižnice musia upravovať obrázok v prehliadači. Testy sme vykonávali s dvoma obrázkami - s obrázkom ABSTRACT (3000 × 2000px; 72dpi; 5,38MB) a s väčším obrázkom PANORAMA (24442 × 4195px; 150dpi; 28,1MB).

http://scottcheng.github.io/cropit/ (A) (30.4.2016)
: je jQuery rozšírením, umožňuje nahrávať obrázky pomocou dragdrop aj vybraním obrázka cez modálne okno. Podporuje približovanie (len pomocou posuvnika) a tiež rotovanie obrázka. Na vykresľovanie obrázku nepoužíva <*canvas*> ale <*img*>, s tým, že keď je potrebné obrázok uložiť prekreslí vybraný výsek obrázka do elementu *canvas* z ktorého výsledok vyexportuje. Aj napriek tomu, že na stránke projektu sa uvádza, že je veľmi rýchle aj pri veľkých obrázkoch, v naších meraniach trvalo nahranie obrázka PANORAMA 26 sekúnd a následné posúvanie nebolo možné, nakoľko sa obrázok prekresľoval niekoľko sekúnd.

http://foliotek.github.io/Croppie/ (B) (30.4.2016)
: je ďalšie jQuery rozšírenie, ktoré úpravu obrázkov. Podporuje približovanie aj stredným tlačidlom myši. Samotná knižnica nepodporuje nahrávanie obrázkov. Umožňuje úpravu obrázkov už nahraných a to takým splôsobom, že ich vykreslí to elementu <*img*>, ktorým umožňuje posúvať a pri uložení prekreslí výrez obrázka do canvasu, ktorý vyexportuje. Vzorové použitie však ukazuje aj prípad, kedy sa knižnica dokáže spracovať nahrané dáta. V tom prípade celý obrázok vykreslí do elementu <*canvas*>. Následne s ním pracuje rovnako ako s v prvom prípade s elementom <*img*>. Nahrávanie obrázku PANORAMA trvalo primerne 13 sekúnd (od odoslanie po zobrazenie obrázka) ale

na jeho plynulé posúvanie bolo potrebné čakať ešte zhruba minútu a pol. Myslíme si, že je to spôsobené tým, že vytvorený element <*canvas*> mal rovnaké rozlíšenie ako originálny obrázok, čo značne predĺžovalo dobu vykreslenia v prehliadači. Vzorové použitie považujeme za jednoduché.

http://andyvr.github.io/picEdit/ (C) (30.4.2016)
umožňuje nahrávanie obrázka cez modálne okno. Obrázok následne vykreslí ako pozadie elementu <*div*>. Priblížovanie mení parametre[1], akými sa toto pozadie zobrazuje. Následné ukladanie prebieha opäť cez prekreslenie do elementu <*canvas*> z ktorého je výsek vyexportovaný. Nahranie obrázku PANORAMA bolo celkom rýchle, avšak posúvanie nebolo možné a priblížovanie trvalo dlhú dobu.

http://codecanyon.stbeets.nl/ (D) (30.4.2016)
toto platené (11$) rozšírenie jQuery umožňuje nahrávanie obrázkov aj cez modálne okno, aj cez *Drag&Drop*. Nahraný obrázok vykresľuje v elemente <*img*>, ktorým umožňuje posúvať a následne uloženie obrázka spôsobí prekreslenie do elementu <*canvas*>. Uloženie je potrebné spraviť ručne. Pri nahratí obrázka ABSTRACT posúvanie mierne sekalo, pri nahratí obrázku PANORAMA nebolo možné.

Tieto vybrané knižnice sme porovnali v nasledujúcich vlastnostiach:

1. Závislosti na knižniciach tretích strán
2. Podpora odosielania obrázka cez formulár (element <*form*>)
3. Nahrávanie viacerých obrázkov
4. Podpora nahrávania cez *DragDrop*
5. Možnosť stanoviť orezanie (napríklad posúvaním obrázka)
6. Podpora priblížovania

1. používajú sa CSS parametre *background-size* a *background-position*

7. Čas nahratia obrázka ABSTRACT
8. Čas nahratia obrázka PANORAMA

	Naša knižnica	A	B	C	D
1	žiadne	jQuery	jQuery	žiadne	jQuery
2	áno	nie	nie	nie	áno
3	áno	nie	nie	nie	nie
4	áno	nie	nie	nie	áno
5	áno	áno	áno	áno	áno
6	áno	áno	áno	áno	áno
7	0,7s	2,78s	1,89s	1,46s	1,85s
8	3,78s	25,99s	min. 15,34s	6,25s	20,49s

Tabuľka 3.1: Porovnanie našej knižnice s inými vybranými riešeniami

Nepodarilo sa nám nájsť riešenie, ktoré by fungovalo obdobne ako naše (používalo po celú dobu len element <*canvas*>). Každé z vybraných riešení však k problematike nahrávania a zobrazovania obrázka pristupuje inak. Žiadne z vybraných riešení nebolo schopné vysporiadať sa so skutočne veľkým obrázkom, zatiaľ čo naše riešenie ho nie len nahralo v rekordne krátkom čase, ale zároveň hneď umožňovalo aj jeho plynulú manipuláciu.

4 Záver

V tejto práci sme najskôr predstavili najpoužívanejšie spôsoby na nahrávanie obrázkov v súčasnosti. Následne sme poukázali na ich nedostatky, ktoré boli základom pre naše riešenie. Stanovili sme si ambiciózne ciele a následne sme naše riešenie vytvárali tak, aby ich spĺňalo. Chceli sme vytvoriť knižnicu, ktorá bude nezávislá od iných knižníc a frameworkov. To sa nám použitím iba JavaScriptu poradilo. Vďaka zapuzdreniu, ktoré poskytujú webové komponenty nie je možné, aby tento JavaScript zasahoval do ostatných častí aplikácie. Takisto implementácia knižnice ako webovej komponenty spôsobuje, že naše riešenie je veľmi jednoduché na použitie. Podporuje nahrávanie obrázkov cez modálne okno, ale aj modálne okno a rovnako podporuje aj nahrávanie vieacerých obrázkov. Všetko toto je možné spraviť aj cez mobilné zariadenia. Obrázky je možné nahrávať či bežnou cestou cez formulár (v elemente <*form*>) alebo cez *XMLHttpRequest* a teda bez refreshu. Užívateľ ale v oboch prípadoch vidí nahrávaný obrázok ešte pred odoslaním a má možnosť upraviť orezanie. Keďže si uvedomujeme, že použitie elementu <*canvas*> na zobrazenie obrázka s možnosťou prístupu až k jednotlivým pixelom otvára dvere obrovskému množstvu možností (napríklad zoomovanie, detekcia tvári, úprava farebnej škaly, inteligentné orezanie), ktorých implementácia nie je v naších silách, vymysleli a popísali sme jednoduchý, zato elegantný spôsob ako tieto rozšírenia písať, použiť a kombinovať a to či už globálne alebo pre konkrétne elementy. V neposlednom rade sme použitie zdokumentovali a s celou knižnicou zverejnili v službe *Github*.

Po predstavedstavení a rozbore nášho riešenia v kapitole 3, sme ho v ďalšej kapitole otestovali a porovnali s vybranými inými riešeniami, ktoré sú založené na podobných princípoch. Na konci tejto kapitoly ešte uvádzame odporúčané použitie a uvažujeme, akým spôsobom by sa mohlo naše riešenie ďalej posúvať.

4.1 Prínosy a odporúčané použitie

Knižnica umožňuje okamžite zobraziť obrázok bez potreby serveru. Všetky zmeny sa dejú v prehliadači a teda skúsenosť používateľa nie

je zaťažená načítavaním. Odporúčame ju používať na projekty, kde je známe finálne rozlíšenie zobrazovaného obrázka (napr. blogy, internetové obchody, firemné stránky, profilové obrázky) a nie je potrebné si uchovávať pôvodný veľký obrázok. Hoc je možné použiť knižnicu aj v takýhto prípadoch, strácajú sa ďalšie prínosy knižnice. A to skutočnosť, že sa odosiela zmenšená a orezaná verzia, ktorá tak menej zaťažuje sieť a v prípade spoplatnených mobilných dát šetrí peniaze používateľa. Tiež sa stráca výhoda, pri ktorej nie je potrebné implementovať zmenšovanie a orezávanie obrázka na serveri. V prípade odosielania obrázka ako *Base64* reťazca (štandardný spôsob) je dokonca množstvo prenášaných dát asi o 30% väčšie.

4.2 Nápady na vylepšenie

Vzhľadom k limitáciam, na ktoré sme počas vývoja narazili dúfame, že našu prácu bude možné v budúcnosti vylepšiť v nasledujúcich oblastiach:

Zrkadlenie časti obsahu Custom Elementu do DOM Ako sme v kapitole 3.4.5 nezhoda medzi tvorcami prehliadačov spôsobuje, že v dobe písania sa už neodporúča na zrkadlenie využiť element <*content*> ale ešte sa neodporúča používať element <*slot*>. Po štandardizovaní štandardu a následnom zapracovaní do knižnice by to umožnilo odosielanie viacerých obrázkov vo formulári (elemente *form*)-

Odosielanie obrázkov ako *Blob* Vzhľadom k tomu, že ukladanie obrázka ako *Base64* reťazec, zväčšuje jeho veľkosť o 30%, ideálne by bolo ukladať ho a prenášať ho ako *Blob*. To je v dobe písania tejto práce podporované len v prehliadači Mozilla Firefox, IE 11 a Google Chrome ohlásil podporu v budúcej verzií (50) [21].

Literatúra

[1] Mozilla Developer Network, "<template>." https://developer.mozilla.org/en-US/docs/Web/HTML/Element/template. cit. 2016-04-13.

[2] http://caniuse.com, "Can I use... Support tables for HTML5, CSS3, etc." http://caniuse.com. cit. 2016-04-13.

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

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

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

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

[7] C. David Goldman, "The beginning of the end for Adobe's Flash." http://money.cnn.com/2011/11/10/technology/adobe_flash/, 11 2011. cit. 2016-04-13.

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

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

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

[11] O. S. ASA, "Opera Software ASA - Opera version history." http://www.opera.com/docs/history/. cit. 2016-04-13.

[12] WebComponents.org, "webcomponents.js." https://github.com/WebComponents/webcomponentsjs. cit. 2016-04-12.

[13] Mozilla Developer Network, "Shadow dom." https://developer.mozilla.org/en-US/docs/Web/Web_Components/Shadow_DOM. cit. 2016-04-13.

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

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

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

[17] K. Finley, "What Exactly Is GitHub Anyway?." http://techcrunch.com/2012/07/14/what-exactly-is-github-anyway/. cit. 2016-04-12.

[18] Mozilla Developer Network.

[19] W. Page, "The state of Web Components." https://hacks.mozilla.org/2015/06/the-state-of-web-components/, year = "2015", month = "6", note = cit. 2016-04-24.

[20] Mozilla Developer Network, "<content>." https://developer.mozilla.org/en-US/docs/Web/HTML/Element/content. cit. 2016-04-24.

[21] Mozilla Developer Network.