VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ROZŠÍŘENÍ MOŽNOSTÍ PREZENTACÍ V PROSTŘEDÍ JUPYTER NOTEBOOK

EXTENDING PRESENTATION FEATURES OF JUPYTER NOTEBOOK

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **PETER PASTOREK**
AUTHOR

VEDOUCÍ PRÁCE **Doc. RNDr. PAVEL SMRŽ, Ph.D.**
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/18728/2016/xpasto07

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Pastorek Peter**
Obor: Informační technologie
Téma: **Rozšíření možností prezentací v prostředí Jupyter Notebook**
Extending Presentation Features of Jupyter Notebook
Kategorie: Umělá inteligence

Pokyny:

1. Seznamte se se stávajícími rozšířeními prostředí Jupyter Notebook, zaměřenými na prezentaci, a specifickými požadavky prezentací v kurzu Skriptovací jazyky na FIT VUT v Brně.
2. Navrhněte a implementujte rozšíření vybraného systému prezentací, který usnadní přípravu prezentací a jejich sdílení.
3. Realizujte rozšíření usnadňující přebírání výkladu konkrétních témat z webových zdrojů a provazování vytvořeného kódu s dokumentací.
4. Vyhodnoťte vytvořený systém při zpracování podkladů k přednáškám předmětu ISJ.
5. Vytvořte stručný plakát prezentující práci, její cíle a výsledky.

Literatura:

- dle doporučení vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- funkční prototyp řešení

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

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

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

Vedoucí: **Smrž Pavel, doc. RNDr., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

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

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Práca sa zaoberá z prostredím Jupyter Notebook. Rozširuje jeho schopnosti zamerané na prezentáciu programovania. Umožňuje preberať materiály z html dokumentov, vylepšuje vzhľad prezentácií, umožňuje automatizovanú prípravu dokumentu na prezentáciu, upravenie nastavenia pre rozšírenia RISE a úpravu konvertovania dokumentu do formátu LaTeX.

Abstract

This work deals with Jupyter Notebook environment. It extends its abilities for presenting programming, allows downloading of materials from html documents, improves the look of presentations, allows automated preparation of the document for the presentation, editing of RISE settings used for presentation, and editing conversion of the document into LaTeX.

Kľúčové slová

Jupyter Notebook, RISE, html, css, javascript, prezentácia programovania, výučby programovania, Python, LaTeX, reveal.js, formát base64

Keywords

Jupyter Notebook, RISE, html, css, javascript, presenting programming, teaching programming, Python, LaTeX, reveal.js, format base64

Citácia

PASTOREK, Peter. *Rozšíření možností prezentací v prostředí Jupyter Notebook*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Smrž Pavel.

Rozšíření možností prezentací v prostředí Jupyter Notebook

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Doc. RNDr. Ph.D. Pavla Smrže. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Peter Pastorek
14. mája 2017

Poďakovanie

Chcel by som sa poďakovať vedúcemu práce pánovi Doc. Pavlu Smržovi za smerovanie pri práci a odbornú podporu.

Obsah

Kapitola 1

Úvod

1.1 Motivácia

Na prezentáciu sa najčastejšie používajú lineárne prezentácie. Najčastejšie sa vytvárajú pomocou programov, ako sú napríklad PowerPoint alebo LaTeX knižnice Beamer. Bežné lineárne prezentácie často nestačia na prezentovanie zložitejších materiálov, ako je napríklad programovanie.

> One reason for the problems with teaching programming could be due to the way it is taught with the traditional lecture format perhaps being inappropriate for the subject. Traditionally lectures for programming have consisted of explanations of concepts and syntax with blocks of related sample code displayed to show how they are used. As this approach is purely theoretical and doesn't show the output of code students may struggle to understand the concepts being explained as they don't see what the code does. However, if code output could be shown on the screen then programming should become more understandable. Research by Lahtinen et al.(2005) suggests this would aid learning as they found via a student survey that learning by doing, where students tried code themselves thus seeing how it works, was seen as more useful than traditional lectures. Therefore, if seeing what code does aids understanding then enhancing lectures with examples of how code works should be beneficial [1].

Preto existujú špeciálne nástroje a prostredie, ktoré umožňujú vytvárať zložitejšie a interaktívnejšie prezentácie.

> Existing presentation tools and document formats show a number of shortcomings in terms of the management, visualisation and navigation of rich cross-media content. While slideware was originally designed for the production of physical transparencies, there is an increasing need for richer and more interactive media types [9].

Pri prezentácií programovania je vhodné do prezentácie vložiť kus programu. Existuje viacero nástrojov na túto funkciu ako napríklad LaTeX knižnica *listings*. Kód vložený knižnicou *listings* môže vyzerať pekne, ale je statický. Pri prezentácií sú potrebné ukážky. Pri knižnici *listings* musia byť všetky varianty zdrojového kódu pripravené pred prezentáciou.

> Web technologies have become an important part of the toolkit, used in any form of education. Nowadays every modern university has a web portal, where a student can find tests and necessary information needed in his study. The portal structures and the kinds of tests and information can be different. Learning management system Moodle [1], for example, provides more than ten file formats.
>
> Faculty members often create training materials on a home desktop using one of the office technology formats (e.g. using Microsoft Office or LibreOffice). With the help of LaTeX document markup language professors can create pdf-documents with mathematical formulas, e.g. MIT educational materials [2]. Some faculty members are able to use the newest web technologies and design easy-to-use hypertext documents and presentations. It has become already popular [3].
>
> As a result such web portals often contain information, which is unstructured and doesn't meets modern requirements. It's difficult to enter and use them from mobile devices and the search documents by text or formula fragments is usually impossible [10].

Na rozšírenie prezentácií je populárna knižnica `reveal.js`. Zobrazuje sa vo webovom prehliadači. Na rozdiel od bežných prezentácií nie je lineárny, ale pracuje v $2D$. Vďaka tomu môžu byť na konci sekcie snímky, ktoré nemusia byť vždy použité. Môžu to byť ilustračné príklady, ktoré sa použijú, ak si to poslucháči žiadajú. Ak poslucháči nepotrebujú viacej ilustračných príkladov, tak môžu byť preskočené jedným kliknutím tlačidla. Použitím html5 na snímkach môže byť čokoľvek od statického textu až po video. Napríklad aj spustiteľný kód. Pretože sa púšťa v prehliadači musí byť napísaný ako html, css alebo javascript. Na použitie iných jazykov je potrebný server. Napríklad aj systém Jupyter. Z rozšírením systému Jupyter sa budem zaoberať práve v tejto práci.

1.2 Návrh

Táto práca sa zaoberá prostredím Jupyter Notebook za účelom jeho rozšírenia. Slúži na vytváranie interaktívnych dokumentov. Na rozdiel od bežných prostredí na vytváranie dokumentov špecializuje sa na kombináciu formátovaného textu a spustiteľného kódu. Podporuje viac ako 40 rôznych jazykov. Pre účeli práce je prevažne najzaujímavejší jazyk Python a jeho výučba použitím systému Jupyter. Systém Jupyter je založený na servere a webovom rozhraní. Server je napísaný v jazyku Python a na webové rozhranie je použitá kombinácia jazykov html, css a javascript. Rozšírenia systému Jupyter, ktoré sú súčasťou tejto práce, sa väčšinou týkajú webového rozhrania.

Budú sa zaoberať:

- Podporou vytvárania dokumentu automatizovaným preberaním zdrojov. Ako primárny zdroj budú webové stránky. Užívateľ musí byť schopný vybrať časť stránky, ktorá bude vložená. Do dokumentu vloží bunky, korešpondujúce časti stánky. Vhodný spôsob vloženia bude cez schránku clipboard.

- Vkladaním obrázku do dokumentu. Prostredie Jupyter nepodporuje obrázky vstavané v dokumente. Obrázky musia byť v externých súboroch a do dokumentu sa vložia použitím odkazu. Ale je možné použiť zakódovanie obrázku na text. Takto zakódovaný obrázok môže byť uložený vo vnútorných dátach dokumentu a odtiaľ odkazovaný.

- Zjednodušením manipulácie s dokumentom. Tá sa dosiahne automatizáciou niektorých činností nad dokumentom. Príklad činností, ktoré sa môžu automatizovať, zahŕňa priraďovanie atribútov bunkám a prípravu dokumentu na použitie.

- Vzhľadom výslednej práce. Na vloženie formátovaného kódu sa používajú bunky typu markdown. Markdown je odľahčený značkovací jazyk. Slúži na pridanie formátovanie jednoduchému textu. Prostredie Jupyter podporuje ďalšie možnosti, ktoré sú použiteľné na rozšírenia jazyku markdown. Ďalšou úpravou vzhľadu je zmenenie písma.

- Podporou pri prezentácií a rozšírením možností prezentácie. Zaoberá sa s rozšírením RISE, ktoré slúži na prezentáciu. Rozšírenie RISE používa na prezentáciu knižnicu `reveal.js`. Mnou vytvorené rozšírenie sa bude zoberať nastavením rozšírenia RISE a knižnice `reveal.js`. Jeho ďalšou súčasťou budú funkcie, ktoré uľahčia manipuláciu s prezentáciou.

- Konverziou dokumentu. Dokument sa dá konvertovať na iné formáty použitím nástroja `nbconvert`. Na konverziu do formátu LaTeX nástroj `nbconvert` používa program `pandoc`. Rozšírenia, ktoré sú časťou tejto práce, sa zaoberajú úpravou a rozšírením konvertovania pomocou nástroja `nbconvert`.

Vytvorené rozšírenia sú znázornené v obrázku 1, aj s ich pôsobením na dokument.

Kapitola 2

Použitie a rozšírenie Nbconvert

Nbconvert je nástroj, ktorý prevádza notebook do iných formátov. Pre účeli tejto práce mňa hlavne bude zaujímať prevod do LaTeX formátu a následne do pdf.

2.1 Jinja šablóny

Konverzia sa dá najjednoduchšie upraviť pomocou Jinja šablón [5]. Tieto šablóny sa píšu vcelku jednoducho, ale nemajú veľkú silu. Dokážu rozšíriť iné šablóny, zväčša natívne. Pracuje na základe blokov.

- Názvy týchto blokov označujú, ktoré časti dokumentu sa budú upravovať.
- V týchto blokoch bude podoba výslednej úpravy.
- V blokoch sa dajú použiť premenné dokumentu a aj podmienky.
- Premenné dokumentu sú väčšinou meta-dáta buniek.

Jinja šablóny sa najčastejšie používajú na skrášlenie výsledného výstupu, alebo odstránenie nechcených informácií.

2.2 Pred-procesory

Ďalšou možnosťou úpravy konvertovania sú pred-procesory [7]. Pred-procesor je Python kód, ktorý dedí z špeciálne vytvorenej triedy. Prepísaním určitých metód môže upravovať logiku konverzie celého dokumentu alebo len niektorých buniek. Existuje všeobecný pred-procesor, ale aj špecifické pred-procesory. Napríklad `LatexPreprocessor`. Pred-procesory sa dajú priradiť vložením do priečniku odkazovaného v `PYTHONPATH` a zaregistrovaním v `jupyter_nbconvert_config.py`.

2.3 Nbconvert knižnica

Nbconvert sa dokáže použiť aj ako knižnica v jazyku Python [6]. Proces použitia tejto knižnice sa dá rozdeliť na dve časti.

- Načítanie dokumentu, ktoré sa zabezpečí použitím knižnice `nbformat`. Vstup je dokument v textovom formáte. Najskôr zaistí, že je dokument korektný. Potom ako

výstup vráti špeciálny slovník, ktorý obsahuje dokument už rozdelený a pripravený na spracovanie.

- Druhá fáza sa skladá z exportovania. Konkrétny použitý exportér záleží od výsledného formátu, ktorý je potrebné dosiahnuť. Mňa bude hlavne zaujímať `LatexExporter`. U exportéra je potrebné ešte pred použitím nastaviť šablónu. Šablóna určuje typ výsledného dokumentu. Napríklad článok alebo reportáž. Výsledok takéhoto exportovania je text už konvertovaného dokumentu a poprípade ďalšie zdroje.

Medzi týmito dvoma časťami môžem dokument, ktorý je v špeciálnom slovníku, slobodne upravovať.

Kapitola 3

Rozšírenia pre dokument

Nbextensions (`jupyter_contrib_nbextensions`) je zoskupenie rozšírení pre Jupyter notebook a systém na ich spravovanie.

> Nbextensions sú uložené ako špeciálne pod adresáre v `src/jupyter_contrib_nbextensions/nbextensions`. Každé rozšírenie notebooku typicky má vlastný adresár obsahujúci:
>
> - `thisextension/main.js` – javascript implementujúci rozšírenie
> - `thisextension/main.css` – voliteľné CSS
> - `thisextension/readme.md` – readme súbor opisujúci rozšírenie v markdown formáte
> - `thisextension/config.yaml` – súbor popisujúci rozšírenie pre `jupyter_nbextensions_configurator`
>
> Volný preklad z [18].

Jupyter Nbextensions Configurator je rozšírenie pre Jupyter notebook. Umožňuje jednoduché povolenie, zakázanie a nastavenie jednotlivých rozšírení [17].

Kapitola 4

Utility pre dokument

Utility pre dokument, ktoré som vytvoril, združujem v rozšírení `jupyter_common_utils`. Združuje funkcie, ktoré pomáhajú pri vytváraní a príprave dokumentu. Upravuje vzhľad, prehľadnosť a automatizuje niektoré časti vytvárania a prípravy dokumentu.

4.1 Príprava dokumentu

Pre rozšírenie RISE, ktoré sa stará o prezentáciu, je potrebné nastaviť typ snímky pri bunke. Typ musí byť nastavený zvlášť pre každú bunku, ale tento systém môže byť rozšírený. Môže byť použité tlačidlo, ktoré nastaví typ snímky bunkám. Typ snímky bude nastavený iba bunkám, ktoré ho nastavený nemajú. Typ snímky sa získa použitím `(cell.metadata.slideshow || {}).slide_type`. Za nenastavený typ snímky sa berie `undefined` alebo `"-"`. Typ snímky, ktorý bude nastavený, môže byť natvrdo zakódovaný na `"subslide"`. Ale je lepšie, aby ho užívateľ mohol zvoliť. Na výber typu snímky je vhodné použiť selektor, pretože ich je konečný počet a musia mať presný tvar. Buď môže byť zadaný pomocou dialógu alebo nastavením rozšírenia v `jupyter_nbextensions_configurator`. Dialóg bude musieť byť vyhotovený ako jquery dialóg. Bude obsahovať popisný text, selektor a tlačidlo. Selektor má možnosti `slide`, `subslide` a `fragment`. Pri otvorení dialógu bude vybraný `subslide`. Po zatlačení tlačidla sa bunkám nastaví vybraný typ snímky a dialóg sa ukončí.

Pred použitím dokumentu, užívateľ môže chcieť niektoré kódové bunky spustené a iné nespustené. Systém Jupyter automaticky ukladá stav dokumentu, preto pri viacerých prezentáciách rovnakého dokumentu jeho stav nemusí byť stály a správny. Ale môže byť vytvorená funkcia, ktorá dokument inicializuje. Inicializácia môže byť spustená tlačidlom. Pri inicializácií sa prekreslia markdown bunky. Spustia sa aj kódové bunky s magickým príkazom `%%html`, pretože sa väčšinou používajú ako alternatíva k markdown bunke. Kódové bunky, ktoré sa majú spustiť, musia mať na začiatku komentár `# Run`. Ostatným kódovým bunkám sa vyčistí výstup.

Opakom inicializačnej funkcie je funkcia na nulovanie. Kde inicializačná funkcia spustí niektoré bunky a iné vynuluje, funkcia na nulovanie vyčistí výstupy všetkým bunkám. Zabudovaná funkcia systému Jupyter môže byť použitá na spustenie všetkých buniek.

Ďalšia funkcia potrebná pre užívateľa je kopírovanie buniek medzi dokumentami. Táto funkcia je už vstavaná vo verzii 5.0 Jupyter dokumentu, ktorá nedávno vyšla. Ako pozostatok skúmania tejto funkcionality, zostala funkcia označenia všetkých buniek. Označené bunky dokumentu môžu byť skopírované do iného dokumentu.

4.2 Kombinácia kódu a odkazu

Zlepšenie výzoru kombinácie kódu a odkazu sa dá riešiť použitím html značiek. Html značky môžu byť vložené kdekoľvek do markdown bunky. K týmto značkám sa dajú priradiť zabudované štýly, ako napríklad `alert`, `alert-success` a `alert-error`. Ak zabudované štýly nie sú dostačujúce, vlastné štýly sa dajú pridať uložením súboru `custom.css` do základného profilu, umiestneného v `~/.jupyter/custom/`. Alebo aj pomocou Python bunky [15].

Pomocou css a html sa dajú urobiť rozsiahle zmeny. Tieto zmeny sa môžu premietnuť aj do doplnku RISE pomocou css modifikátoru `!important`.

Ich nevýhodou je veľká réžia. Ich výsledná forma nie je vždy intuitívna, preto je k nim potrebné vytvoriť alternatívu. Ako danú alternatívu som vybral zápis `{c{example}http://www.example.com/reference }`. Vybral som ho predovšetkým, lebo značky { a } neboli v základnom notebook-u použité. Javascript, patriaci k rozšíreniu, pri vykresľovaní (rendering) markdown bunky vyvolá akciu. Mnou zvolený zápis `{c{example}http://www.example.com/reference }` prepíše na HTML značky. Toto prepísanie sa rieši cez regulárne výrazy.

Pri práci sa však naskytol jeden zvláštny problém. Moje konvertovanie sa spúšťa až po základnom konvertovaní. To väčšinou nie je problém. Až na to, že textové odkazy, ktoré sú použité, môžu byť prepísané do HTML značiek. Potom môj zápis `{c{example}http://www.example.com/reference }` mohol vyzerať ako `{c{example}<a target="_blank" href="http://www.example.com/reference">http://www.example.com/reference</a> }`. Táto situácia musela byť riešená použitím viacerých regulárnych výrazov v správnom poradí.

Pridal som aj zápis `{ci{example}#bookmark }`, s „i“ ako internal. Tento zápis je vhodný použiť na odkazy do používaného dokumentu.

Mnou vytvorené značky sú dobré, keď sa dokument píše. Nie sú však vhodné, keď je dokument už vytvorený alebo keď je vytváraný automatizovaným nástrojom. Napríklad automatizovanom vkladaní html stránky do dokumentu. Preto som sa rozhodol použiť originálnu kombináciu značiek `[‘text‘](url)` v kombinácií so mnou vytvoreným vykreslením. Keďže moje konvertovanie sa spúšťa až po základnom, stačí k odkazu `<a ... >` pridať mnou vytvorenú triedu, ak obsahuje značku `<code>`. Odkaz `<a ... >` nemusí obsahovať iba značku `<code>`. Kombinácia značiek `[**‘text‘**](url)`, ktorá text zvýrazní, je tiež dovolená. A je potrebné ju brať do úvahy.

Na kombináciu kódu a odkazu podčiarknutie nie je vhodné. Zmena farby písma môže byť na tento účel vhodná. Zmena pozadia podľa mňa nie je vhodná, pretože čitateľ to môže považovať za niečo iné ako kód. Vhodné riešenie môže byť aj ohraničenie. Ohraničenie napodobňuje tlačidlo. Ak bude jeho pozadie a font rovnaký ako kód, tak čitateľ ho môže považovať za kód a je na neho možné kliknúť. Problém s použitím ohraničenia je, že tlačítka sa nehodia uprostred textu. Vyzerajú vyňaté z textu. Preto sa súvislý text môže ťažko čítať. To sa ale dá vyriešiť použitím iba horného a dolného ohraničenia. Pretože nie je ľavé a pravé ohraničenie, súvislý text sa dá jednoduchšie čítať. Týmto spôsobom má odkaz vyzerať ako tlačidlo v súvislom texte. Ďalší problém tohto spôsobu je s prekrývaním. Ak sú odkazy v texte nad sebou, tak sa ich ohraničenie bude prekrývať. Alebo môže byť príliš tenké. Môže byť nastavené, aby sa neprekrývali pomocou vypchávky (padding). Potom budú medzery medzi riadkami väčšie v riadkoch, ktoré majú kombináciu odkazu a kódu. Mať nerovnomerné riadkovanie nie je vhodné. Prekrývanie sa môže vyriešiť aj pomocou zväčšenia medzier medzi riadkami. Riadkovanie je rovnomerné. Ale nie v každom dokumente alebo bunke je použitá kombinácia odkazu a kódu. Zväčšením riadkovania sa zmenší množstvo informácií na plochu. Ohraničenie ani nemusí fungovať s každým fontom. Napríklad pri

fonte Roboto Mono sa kód v odkaze roztiahne a zakryje časť ohraničenia. V LaTeX sú odkazy modré bez podčiarknutia a vyzerajú dobre. Preto som napodobnil ich vzhľad pre Jupyter dokument. Ale ako farbu som použil základnú farbu pre odkazy.

Kód sa zvýrazňuje špeciálnym fontom a sivým pozadím. Ak je kód odkaz, tak má modrú farbu. Ak nie je kód odkaz, tak má farbu paragrafu, ktorá je natívne čierna. To, že na odkaz je možné kliknúť, sa znázorňuje zmenou kurzora a zmenou odtieňa modrej farby, ktorá sa používa na odkazy. Okrem sivého pozadia kódovej bunky sú použité natívne farby.

Mnou vytvorené značky sa pri konverzií natívne neprekladajú. Tento preklad je potrebné zabezpečiť. Mal by byť spúšťaný stále a aj bez špecifického určenia užívateľom. Preto bol na túto úlohu zvolený pred-procesor. Pred-procesor môže byť nastavený, aby sa spúšťal pri každej konverzií. Mnou vytvorená značka môže byť konvertovaná na rôzne tvary. Použil som tvar markdown odkazu. Predpokladám, že nástroje na jeho konverziu budú fungovať najlepšie. V pred-procesore sa dá zistiť cieľový formát konvertovania. Konvertovanie do formátu LaTeX je možné špecializovať. Natívne je na kód v markdown bunke použité strojové písmo. Na odkazy je použité modré písmo bez podčiarknutia. K strojovému písmu som ešte pridal sivé pozadie, aké sa používa v niektorých vývojových nástrojoch. Sivé pozadie neprekáža modrému písmu odkazu.

4.3 Písmo na programovanie

Natívne strojové písmo, ktoré prostredie Jupyter používa, nie je vhodné. Strojové písmo sa najčastejšie používa na v prostredí Jupyter na zobrazenie kódu. Keď boli dve podčiarknutia vedľa seba, nebolo viditeľné, kde jedno začína a druhé končí. Preto som nastavil veľkosť písma z 100% na 80% a vzdialenosť medzi písmenami z 25% na 30%. Toto nastavenie funguje, ale nie je optimálne. Rozširuje všetok kódový text v markdown bunke. Lepšie riešenie je zmeniť font. Je potrebné vybrať kvalitný programovací font, ktorý bude mať skrátenú pod-čiarku. Táto požiadavka nie je populárna. Nedajú sa vyhľadávať písma, ktoré túto požiadavku spĺňajú. Našiel som 3 fonty, ktoré túto požiadavku spĺňajú: Hermit, Inconsolata a Roboto Mono. Font Hermit vyhovoval požiadavkám, ale nedá sa pridať z internetu. Musí byť stiahnutý a pridaný k programu. Dá sa použiť zadarmo, ale k programu musí byť priradený aj s open font licenciou. K bakalárskej práci sa dajú pridávať licencie, ale použitím google fontu to nie je potrebné, pretože font nie je súčasťou práce, ako napríklad Inconsolata. Je to populárny monospace google font. Dá sa pridať použitím linku na google css stylesheet. Aj keď dve pod-čiarky sa dajú rozoznať v normálnom zobrazení, ale v prezentácií pomocou RISE medzeru nie je dobre vidieť. Font Roboto Mono je google font, tak isto ako Inconsolata. Ale medzera medzi pod-čiarkami je výraznejšia. Dobre je vidieť v normálnom zobrazení, ako aj v prezentácií RISE. Font som nastavil aj pre kódové bunky. Ako na vstup, tak aj na výstup.

Kapitola 5

Porovnanie kódu

Počas prezentácie sa veľmi dobre ukazuje rozdiel v kódoch prepínaním snímok bez prechodu. Časti, ktoré sú rovnaké, sa nehýbu ale časti, ktoré sú rozdielne, áno. Človek si najlepšie všíma náhlu zmenu a pohyb. Okamžitý prechod medzi bunkami sa dá nastaviť v rozšírení *RISE configuration*.

5.1 Rozlíšenie kódu, ktorý má byť porovnaný

Iba niektoré kódové bunky sa budú porovnávať, preto je ich najskôr potrebné oddeliť od ostatných. Automatizované rozpoznanie nie je dostatočne spoľahlivé. Spoľahlivejšie je označenie buniek užívateľom.

Ako prvé a to najjednoduchšie ma napadlo, na prvý riadok kódu dať určitý zvolený komentár. Napríklad `# Compare code`. Keď používam `nbconvert` ako knižnicu, tento komentár dokážem jednoducho zachytiť a spracovať dané snímky podľa požiadaviek. Tento spôsob nie je elegantný na použitie ani estetický, ale slúži dobre ako dočasné riešenie alebo na odladenie súčastí programov.

Chcel som daný systém rozšíriť a to použitím prvkov Jupyter notebooku. Ako najvhodnejší prvok pre dané rozšírenie som považoval Cell toolbar [3]. Je to prvok zobrazujúci sa v záhlaví buniek. Bežne sa využíva na vybranie štýlu snímky pre rozšírenie RISE. Týmto prvkom sa nastavujú metadata snímky. Tieto dáta sú potom prístupné v rozšíreniach, napísaných v javascripte, aj cez `nbconvert` v Python-e. Pri nastavení tohoto prvku sa dá vybrať, pri ktorých bunkách sa bude toolbar zobrazovať. Problém nastane v tom, že sa dá zobraziť naraz len jeden toolbar. Prepínajú sa v hlavnej ponuke cez View a Cell toolbar.

5.2 Ako zobraziť komparáciu v prostredí LaTeX?

Často na predstavenie určitého princípu sa používa porovnanie daného princípu s podobným alebo opačným. Tak isto sa môže prezentovať aj kód a jeho výsledky. Preto nie je nezvyčajné, keď sú za sebou dve bunky, ktorých kód sa líši len v jednom alebo dvoch riadkoch a ich výstupmi.

Pri zobrazení daných buniek pod sebou ich rozdiel nie je na prvý pohľad zrejmý. Dali by sa zobraziť aj vedľa seba, zarovnané na riadky. Ale takto by sa buď nezmestili, alebo by boli veľmi malé. Rozhodol som sa ich dať do jednej bunky, kde spoločné riadky budú len raz a rozdielne riadky budú oba. Pričom musia byť od seba rozoznateľné.

Ako prvé testovacie riešenie som zvolil identifikátory pred rozdielnymi riadkami. Napríklad `A:` alebo `B:`.

Za lepšie riešenie som považoval rozpoznávanie uložiť do pozadia, napríklad zafarbením riadku. Systém, ktorý nbconvert používa, je neupraviteľný. Preto som sa rozhodol zmeniť bunky na typ raw a použiť prostredie `listings` [19]. Prostredie `listings` umožňuje použitie escape znakov. Použitím escape znakov môžem do prostredia `listings` vložiť LaTeX značky, ktoré nebudú prostredím `listings` spracované. Týmto som dokázal rozlíšiť riadky [14], aj upraviť číslovanie [21]. Riadky, ktoré sú rozdielne, majú rôznu farbu pozadia a rovnaké číslo riadku.

Prostredie `listings` sa dá použiť zmenením typu bunky na raw. Bunky typu raw nie sú exportérom upravené. Používajú sa niekedy na prepašovanie LaTeX kódu priamo z dokumentu. LaTeX kód pri použití prostredia z balíčku `listings` sa dá jednoduchšie čítať. Obsah kódovej bunky stačí zabaliť do prostredia `listings`. Pre jeho správny výzor aj vo výslednom pdf dokumentu je potrebné toto prostredie správne nastaviť pre jazyk Python. Syntax Python-u už v balíčku je, len je potrebné nastaviť výsledné farby. Doplniť balíky sa dá po exportovaní najlepšie pred príkazom `\begin{document}`.

Kapitola 6

Nastavenia pre RISE

RISE je jedno z rozšírení, ktoré dokáže prezentovať Jupyter dokument. Rozšírenie RISE pracuje na základe `reveal.js` [2]. Systém `reveal.js` je založený na html, css a javascript. Umožňuje prezentovanie v internetovom prehliadači bez použitia externého programu. Preto je pre Jupyter notebook, ktorý pracuje v internetovom prehliadači, vhodný. Pre systém `reveal.js` je potrebný html dokument v špecifickom formáte. Jupyter dokumenty nie sú na túto štruktúru postavené. Rozšírenie RISE zabezpečuje správne prostredie pre `reveal.js` a dočasnú konverziu Jupyter dokumentu do formátu pre `reveal.js`.

Rozšírenie RISE nesprístupňuje všetky funkcie `reveal.js`, preto som vytvoril rozšírenie, ktoré pracuje ako nadstavba pre rozšírenie RISE. Musí byť schopné sprístupniť funkcie `reveal.js` a aj vytvorí viacero ďalším možnosti pre prácu s rozšírením RISE.

6.1 Globálne nastavenia

RISE berie nastavenie z viacerých zdrojov. Jeden zo zdrojov je v meta-dátach dokumentu. Objekt `livereveal` obsahuje položky, ktoré sa použijú na nastavenie RISE. Meta-dáta sa najjednoduchšie nastavia použitím javascriptu. Nastaviť sa môžu použitím zložitejšieho dialógu, alebo kódovej bunky s magickým príkazom `%%javascript`. Po zatlačení tlačidla sa do dokumentu vloží kódová bunka, ktorá obsahuje potrebné príkazy na nastavenie. Príkazy budú formátu `IPython.notebook.metadata.livereveal["%item%"] = %value%;`.

Ako prvky nastavenia môžu byť:

- `theme` – Zmení zobrazenie snímok. Napríklad farba písma a podfarbenie. Základná hodnota je `"simple"`.
- `transition` – Nastaví prechody medzi snímkami. Základná hodnota je `"linear"`.
- `scroll` – Umožní skrolovať snímku. Základne je vypnuté `false`.
- `start_slideshow_at` – Určí začiatok prezentácie. Základná hodnota je `"beginning"`. Ďalšia hodnota je `"selected"`, pri ktorej prezentácia začne na vybranej bunke.
- `controls` – Rozhoduje či sa zobrazia kontrolné prvky. Základne je zapnuté `true`.
- `progress` – Rozhoduje či sa zobrazí progres prezentácie. Základne je zapnuté `true`.
- `history` – Rozhoduje či sa zobrazí história prezentácie. Základne je zapnuté `true`.
- `slideNumber` – Rozhoduje či sa zobrazí číslo snímky. Základne je zapnuté `true`.

- `width` a `height` – Nastavia šírku a výšku snímok. Základné sú nastavené na `1140` a `855`.
- `minScale` – Základne hodnota je `1.0`. Nastavenie je potrebné pre správnu funkciu podprogramu `codemirror`, ktorý sa stará o zobrazovanie kódu v bunke.

Vygenerované nastavenie musí byť upraviteľné. Najlepší spôsob upravenia atribútov pre rozšírenie je použitie parametrov pre `jupyter_nbextensions_configurator`. Bude použitý `list`, kde prvky budú typu `text`. Budú formátu `%item% = %value%`. Týmto spôsobom užívateľ môže pridať vlastné prvky, odstrániť prvky a upraviť základné prvky použitím grafického rozhrania zabudovaného do systému Jupyter. K nastaveniu je vhodné pridať popis nastavenia a možné hodnoty. Keďže sú nastavenia upraviteľné, tak aj popisy k nim musia byť upraviteľné. Aj pre nich môže byť použitý `jupyter_nbextensions_configurator`. Bude použitý `list`, kde prvky budú typu `text`. Prvky budú mať formát `%item% = %description%`. Popisy `%description%` sú jednoriadkové, ale `\n` bude označovať nový riadok a `\t` bude označovať tabulátor. Ak prvok `%item%` má popis, tak sa priradí pred neho. Ak prvok `%item%` nemá hodnotu `%value%`, tak nebude zobrazený ani keď má popis `%description%`.

Nastavenie bude vložené do dokumentu pri jeho spustení, ak nemá vlastné nastavenie. Pre zmenu nastavenia v dokumente existujú dve možnosti. Buď sa nastavenia v bunke vygenerujú z lokálneho nastavenia dokumentu v meta-dátach, alebo z globálneho nastavenia v rozšírení. Obidve možnosti sú potrebné. Preto každá možnosť bude mať vlastné tlačidlo. Jedno na vyvolanie nastavenia z meta-dát dokumentu, druhé na vyvolanie globálneho nastavenia z rozšírenia.

6.2 Vkladanie atribútov jednotlivým snímkam

Rozšírenie RISE nemá zabudované možnosti na manipuláciu parametrov pre jednotlivé snímky. Dalo by sa to zakódovať priamo do rozšírenia RISE, ale takéto riešenie nie je jednoducho prenositeľné a flexibilné. Preto je lepšie vyrobiť vlastné rozšírenie schopné upraviť výsledný dokument. Najskôr je potrebné zachytiť moment po konverzii, kedy je dokument pripravený na prezentáciu. Po ukončení konverzie, RISE pridáva triedu `reveal_tagging` prvku s identifikátorom `maintoolbar`. Pridanie triedy sa nedá natívne zachytiť, ale dá sa vyrobiť. Upravením vstavanej funkcie na pridanie triedy. K jej originálnej činnosti vytvorí udalosť funkciou `$(this).trigger('addClassEvent', arguments)`, ktorá sa už dá zachytiť funkciou `$('#maintoolbar').bind('addClassEvent', reaction)`.

Identifikátor snímky sa skladá z kľúčového slova `slide`, čísla snímky a čísla pod-snímky, napríklad `slide-1-4`. Pri načítaní RISE prezentácie sa prejde všetkými bunkami dokumentu a pritom sa počíta číslo snímky, číslo pod-snímky a číslo fragmentu. Snímkam, ktoré majú zadaný špeciálny atribút, sa pridajú potrebné atribúty. Nájdu sa použitím jquery selektoru `#slide-%slide_number%-%sub_slide_number%`.

Pre snímku sa dá nastaviť parameter `data-transition`. Určuje štýl prechodu snímky. Bude nastavený pomocou rolovacieho zoznamu. Môže byť nastavený žiadny prechod `none`, lineárny `linear`, plynulé zmiznutie `fade`, posunutie `slide` alebo priblíženie `zoom`. Ak nebude nastavený, použije sa globálne nastavenie.

Alebo sa dá nastaviť rýchlosť prechodu `data-transition-speed`. Môže byť použité základné nastavenie `default` trvajúce 800 ms, rýchle nastavenie `fast` trvajúce 400 ms alebo pomalé nastavenie `slow` trvajúce 1200 ms.

Je potrebné nastaviť viacero atribútov, ale v paneli nástrojov pre bunky sa dá nastaviť iba jeden atribút. Preto je ich potrebné vytvoriť niekoľko. Bude vytvorený použitím funkcie

`IPython.CellToolbar.utils.select_ui_generator(choices, setter, getter, name)`. Prepínanie medzi panelmi sa robí pomocou `View -> Cell Toolbar -> Toolbar`. Na to je ale potrebné viacero klikov. Na prepínanie sa môže vytvoriť skratka. Pomocou tlačidla sa môže cyklicky prepínať medzi panelmi nástrojov.

6.2.1 Okamžité prechody medzi snímkami

Systém `reveal.js` má možnosť určovať rýchlosť prechodu pre jednotlivé snímky [4]. Má základné, rýchle a pomalé nastavenie. Nemá nastavenie na okamžitý prechod. Toto nastavenie som si dokázal jednoducho vyrobiť. Rýchlosť prechodu je určená časom trvania transformácie. Preto som vytvoril nové nastavenie s časom trvania transformácie 0 ms. Nastavenie sa aplikuje atribútom `data-transition-speed` na hodnotu `instant`.

Okamžitý prechod sa dá nastaviť nie len rýchlosťou prechodu ale aj cez štýl prechodu. Dá sa nastaviť žiadny štýl prechodu. Žiadny štýl prechodu spôsobí prechod bez transformácie. Nastavuje sa atribútom `data-transition` na hodnotu `none`.

6.2.2 Schovanie častí bunky pri prezentácií

Schovanie bunky v prezentácií RISE je špeciálny prípad pridania atribútu. Základne nie je schovaný žiadny element. Môže byť schovaný vstup `Input` alebo výstup `Output`. Upravuje sa css elementu použitím atribútu `style`. Atribút `style.display` bude nastavený na `"none"`. Nebude nastavený na element snímky, ale na element v nej. Element, ktorému bude daný atribút nastavený, sa vyberie podľa vstupu. Ak je vstup `Input`, bude schovaný element vstupu pre bunku. Ak je vstup `Output`, bude schovaný element pre výstup bunky. Elementy budú nájdené pomocou jquery selektoru `#slide-%slide_number%-%sub_slide_number% div.cell div.input` pri schovaní vstupu `Input` a `#slide-%slide_number%-%sub_slide _number% div.cell div.output` pri schovaní výstupu `Output`. Daná situácia platí len pri bunke typu `slide` a bunke typu `subslide`. Pri bunke typu `fragment` sa do selektoru vloží označenie fragmentu `div[data-fragment-index="%fragment_number%"]` medzi `#slide-%slide_number%-%sub_slide_number%` a `div.cell`. Fragment sa vytvorí neskôr ako snímka. Preto sa počká pomocou funkcie `setTimeout()` dve sekundy, než sa atribút aplikuje.

Pretože sa neupravuje snímka, ale vstup a výstup bunky, musí byť pri ukončení prezentácie odstránené. Pri ukončení prezentácie sa prejde všetkými bunkami. Ak má bunka nastavené schovanie časti bunky, tak sa najskôr označí pomocou funkcie `cell.select()`. Označená bunka sa dá nájsť použitím jquery selektoru `div.selected`. Podľa nastavenia sa vyberie element `div.selected div.input` pri vstupe `Input` alebo `div.selected div.output` pri vstupe `Output`. Chovanie bude zrušené nastavením atribútu `style.display` na `null`.

6.3 Návrat na vybranú bunku

Po ukončení RISE prezentácie sa dokument vráti na začiatok a odznačí sa označená bunka. Bolo by dobré, ak by sa po prezentácií dokument vrátil na miesto, odkiaľ bola ukončená. Podobným systémom ako sa zachytilo spustenie prezentácie sa dokáže zachytiť ukončenie prezentácie. Namiesto pridania triedy `reveal_tagging` k prvku s identifikátorom `maintoolbar` bude trieda `reveal_tagging` odstránená. Pred zrušením prezentácie môže byť zachytená vybraná bunka príkazom `Jupyter.notebook.get_selected_cell()`. Potom je potrebné počkať na skončenie zrušenia prezentácie. Bunka bude označená príkazom `cell.select()`

a okno sa na ňu posunie príkazom `cell.focus_cell()`. Kde `cell` je bunka, ktorá bola získaná pred zrušením prezentácie. Problém je, že bunka nie je vždy označená. Ak táto metóda má fungovať, tak užívateľ pred ukončením prezentácie musí bunku označiť. Ďalšia možnosť je získať identifikátor zobrazenej snímky. Snímka, ktorá je zobrazená, má na rozdiel od ostatných triedu `present`. Zo získaného identifikátora sa dá získať číslo snímky a číslo pod-snímky. Po ukončení zrušenia prezentácie sa musí nájsť správna bunka, ktorá bola na v danej snímke. Najjednoduchšie sa získa prejdením všetkých buniek a počítaním čísel snímky a pod-snímky. Keď sa dostane k získaným číslam, našla sa správna bunka. Tá bude potom označená. Ak je na snímke viacero buniek použitím fragmentov, tak bude vybraná tá prvá. Tým pádom nemôže byť vybraný fragment, ale bude vybraná buď bunka s typom snímka alebo pod-snímka.

Kapitola 7

Automatizovaná tvorba dokumentov

Účelom tohto je vytvoriť rozšírenie, ktoré dokáže automatizovať preberanie materiálov. Musí previesť web stránku, ktorá je daná jej url kódom. Ak je systém externý, tak má vytvoriť nový dokument. Ak je systém interný, tak vloží bunky do otvoreného Jupyter dokumentu. Príklad interného systému je rozšírenie nbextension. Príklad externého systému je skript.

7.1 Priame rozšírenie nbextension

Tento systém by mal byť čisto riešený v javascripte. Môžeme použiť javascript s rozšírením jquery. Kombinácia javascriptu a jquery má veľa a dobré nástroje na prácu s html stránkami a DOM (Document Object Model, objektový model dokumentu). A nbextensions majú prístup ku knižniciam, ktoré pracujú s Jupyter dokumentom. Použitím týchto knižníc sa dá vytvoriť bunka a vložiť ju do dokumentu.

Problém vzniká pri práci so stránkami z inej domény. Bezpečnostný protokol web prehliadačov Same Origin Policy bráni v komunikácií. Zabraňuje aby stránky sledovali užívateľovu aktivitu na stránkach s iným pôvodom. Pôvod sa definuje ako kombinácia protokolu, hostiteľa a portu. Dokumenty s iným pôvodom sú preto izolované. Stránky, z ktorých je potrebné prevziať materiály, majú iných hostiteľov ako server pre Jupyter Notebook [11].

Nenašiel som dosť všeobecné riešenie, ktoré dokáže spoľahlivo Same Origin Policy obísť.

7.2 Externý skript v jazyku Python

Jupyter dokumenty nepoužívajú špeciálne formátovanie ani šifrovanie. Preto na ich vytváranie netreba špecializované nástroje. Jedná sa o `json` dokument so špeciálnym ustanoveným formátom. Jupyter dokument môže byť vo vnútri skriptu reprezentovaný ako stromová štruktúra skladaná zo slovníkov, listov a dát. Koreňový uzol sa dá previesť na reťazec pomocou funkcie `dumps` knižnice `json`. Vygenerovaný reťazec je výsledný Jupyter dokument. Funkcia `dumps` dokáže vygenerovať daný reťazec aj pekne. Nastavením voliteľného parametru `indent` budú mať jednotlivé riadky správne logické odsadenie od ľavého kraja. Odsadenie je určené vzdialenosťou od koreňa stromu.

7.2.1 Získanie html dokumentu

Rozhodol som sa pracovať z html dokumentom pomocou knižnice `lxml.html`. Táto knižnica má funkciu na získanie dokumentu `lxml.html.parse(url)`. Stiahne dokument na zadanej url adrese a pripraví ho na spracovanie. Použitie jednej knižničnej funkcie na stiahnutie a aj spracovanie dokumentu do formy, ktorá sa dá ihneď použiť, znie dobre. Lenže táto metóda nefunguje vo všetkých prípadoch. Niektoré stránky nenájde aj napriek tomu, že ich nájde bežný webový prehliadač. Preto je potrebné použiť silnejšiu funkciu na stiahnutie potrebnej stránky. Ďalší možný kandidát bola funkcia `urllib.request.urlopen(url)`. Táto funkcia nepreukázala problémy spojené z funkciou `lxml.html.parse(url)`. Stránka získaná touto metódou sa dá do štruktúry z knižnice `lxml.html` previesť funkciou `lxml.html .fromstring(document)`. Po použití tohoto postupu je dokument pripravený na spracovanie.

Väčšinou sa očakáva práca so stránkami na webe, ale sú výnimočné prípady, kedy to nie je vhodné. Občas používateľ bude chcieť spracovať html dokument umiestený na lokálnom disku. Keďže bola použitá na konvertovanie funkcia `lxml.html.fromstring(document)`, stačí zmeniť len získanie dokumentu vo forme reťazca. Pri otváraní lokálneho súboru je vhodné uviesť aj znakovú sadu, ktorou bol dokument zakódovaný. Ak nie je zadaná, tak sa zoberie kódovanie UTF-8.

> V štandarde HTML 4 sa používa znaková sada ISO-8859-1. Tá je ale limitovaná. V štandarde HTML 5 sa prešlo na znakový sadu UTF-8. UTF-8 obsahuje takmer všetky symboly na svete. Namiesto ISO-8859-1 sa občas používa aj ANSI, sú identické až na to že ANSI obsahuje o 32 znakov viac [12].

Ďalším možným vstupom je `sys.stdin`. Je ho možné použiť napríklad pri linux rúrach. Pri použití linux rúr daný html dokument môže byť získaný akýmkoľvek programom a presmerovaný priamo do programu.

7.2.2 Prehľadávanie dokumentu

Rozličné html dokumenty môžu mať aj rozličnú štruktúru. Materiál, ktorý nás zaujíma, sa môže nachádzať v iných značkách a vetvách DOM stromu. Je takmer nemožné vyrobiť systém, ktorý bude dosť všeobecný, aby pracoval s väčšinou stránok. Pre uľahčenie tvorby a údržby viacerých stránok je vhodné použiť šablóny. Systém, ktorý pracuje len zo stránkami s určitou šablónou, je možné vyrobiť oveľa realistickejšie. Systém na konverziu môže byť jednoduchší a rýchlejší. Je ale použiteľný len na určité typy stránok. Treba preto časť systému vyňať. Tá sa bude prispôsobovať rôznym šablónam. Šablónu, ktorá má byť použitá, je potrebné identifikovať. Skupiny stránok s rovnakou šablónou môžu byť rozpoznané podľa ich domény. Doména identifikuje miesto uloženia stránky. Stránky na rovnakom mieste, s rovnakým účelom a s rovnakým vydavateľom pravdepodobne budú tvorené podľa rovnakej šablóny. Pri väčších doménach sú pravdepodobne vytvárané strojom. Ak na vyhľadávanie bude použitý systém `xpath`, môže byť podľa domény vybraný reťazec systému `xpath`. Systém na spracovanie nemusí pracovať s celým dokumentom, ale len z jeho časťami.

K jednotlivým častiam dokumentu sa dá pristupovať pomocou funkcie `xpath(string path)`. Použitím systému `xpath` sa dajú vyhľadávať html značky:

- Podľa ich mena. Napríklad `/p` nájde značky `<p>`.

- Podľa ich pozície v dokumente oproti koreňu a ostatným značkám. Napríklad `/p/strong` nájde značky `<strong>` v značke `<p>`. Alebo `//a` nájde značky `<a>` kdekoľvek v dokumente.
- Podľa ich atribútov. Napríklad `//div[@class='post-text']` nájde značky `<div>` z triedou post-text.
- Špeciálny znak `*` môže byť použitý ako zástupca pre všetky značky. Napríklad `//div [@class='post-text']/*` nájde všetky značky v značke `<div>` s triedou post-text.
- Špeciálny znak `|` slúži ako zjednotenie. Na nájdenie použije jednu definíciu alebo druhú. Napríklad `//div[@class='post-text']/* | //span[@class='comment-copy']` nájde značky `<div>` s triedou `post-text` a značky `<span>` s triedou `comment-copy`.
- Špeciálny reťazec `not` môže byť použitý pri atribúte na jeho negáciu. Napríklad `//div [@class='section']/*[not(name()='dl')]` nájde všetky značky v značke `<div>` s triedou section okrem značiek `<dl>`.
- Špeciálne reťazce `or` a `and` môžu byť použité pri atribútoch ako logické operácie. Napríklad `//div[@class='section']/*[not(@class='section') and not(name()= 'dl')]` nájde všetky značky v značke `<div>` s triedou section okrem značiek `<dl>` a značiek s triedou section.

Stránky zo stackoverflow.com majú komplexnú štruktúru, ale nezaujíma nás ich celá štruktúra. Príspevky sa nachádzajú v špeciálnych častiach a tabuľkovej štruktúre. Na konvertovanie je zaujímavý text príspevku. Aj napriek rozdielnej štruktúre medzi otázkou a odpoveďami. Aj keď má každý príspevok špeciálne číslo. Text príspevku sa vždy nachádza v značke `<div>` s triedou post-text. Tie sa dajú nájsť pomocou `//div[@class='post-text'] /*`. Bolo by vhodné do dokumentu uviesť aj komentáre príspevkov. Tie sa nachádzajú v ďalších častiach s špeciálnym číslom a tabuľkách. Avšak ich text sa vždy nachádza v značke `<span>` s triedou `comment-copy`. Tie sa dajú nájsť pomocou `//span[@class='comment-copy']`. Ak sa bude `//div[@class='post-text']` a `//span[@class='comment-copy']` vyhľadávať samostatne, tak najskôr budú údaje z jedného vyhľadávanie a potom údaje z druhého. No komentáre k príspevku by mali byť pod príspevkom, ku ktorému patria. Ale ak sa zjednotia pomocou znaku `|` na `//div[@class='post-text']/* | //span[@class= 'comment-copy']`, tak budú údaje spracované postupne zhora na dol. Budú prichádzať bez ohľadu, ku ktorému reťazcu patria. Komentáre sú v html kóde pod príspevkom, ku ktorému patria, a nad nasledujúcim príspevkom. Preto aj pri rovnoprávnom spracovaní nájdených údajov bude konštrukcia výsledného dokumentu pozostávať z otázky a komentárom k nej, nasledovaná odpoveďami aj s ich komentármi. Bolo by ešte aj vhodné na začiatok dokumentu vložiť aj hlavičku. Ako hlavička dokumentu môže byť použitý titul stánky `//title`, alebo aj značka `//div[@id='question-header']`.

Pre stránky z docs.python.org je text stránky umiestnený v značke `<div>` s triedou `section`. Tá sa dá nájsť pomocou `//div[@class='section']/*`. Stránka sa však delí na úvod a pod-časti, ktoré sú dlhé a sú to tiež značky `<div>` s triedou `section`. Ak by to ostalo takto, tak by boli pod-sekcie v priveľkých snímkoch za úvodom a hneď za tým aj zvlášť rozkúskované. Po úprave `//div[@class='section']/*[not(@class='section')]` budú zdvojenia odstránené. Pod-sekcia nebude v jednej snímke, ale bude rozkúskovaná. Stránka používa zoznamy na zobrazenie funkcií, html značka `<dl>` so značkami `<dt>` a `<dd>`.

Definícia funkcie môže byť príliš dlhá na jednu snímku. Preto je ich potrebné rozdeliť. Najskôr treba zoznamy `<dl>` odstrániť z vyhľadávania. Napríklad pridaním `not(name()='dl')` k `//div[@class='section']/*[not(@class='section')]`. Vznikne z toho `//div[@class='section']/*[not(@class='section') and not(name()='dl')]`. A pridať vyhľadávanie na prvky zoznamu. Nestačí napísať len `//dt | //dd`. Vyhľadávanie by malo obsahovať aj vyhľadávanie sekcie. Pravdepodobne by to fungovalo, aj keby tam vyhľadávanie sekcie nebolo, ale s ním je to bezpečnejšie. Text hlavičky funkcie je priamo v značke `<dt>`. Preto reťazec na jej vyhľadávanie bude `//div[@class='section']/*[not(@class='section')]/dt`. Text tela funkcie je v paragrafoch umiestnených v značke `<dd>`. Preto reťazec na ich vyhľadávanie bude `//div[@class='section']/*[not(@class='section')]/dd/*`. Týmto sa jednotlivé paragrafy rozdelia do samostatných buniek. Výsledná podoba reťazca bude `//div[@class='section']/*[not(@class='section') and not(name()='dl')] | //div[@class='section']/*[not(@class='section')]/dt | //div[@class='section']/*[not(@class='section')]/dd/*`.

Pre stránky z wiki.python.org je text umiestnený v značke `<div>` s identifikátorom `content`. Ako hlavička môže byť použitá značka `<title>`. Preto výsledná podoba vyhľadávacieho reťazca bude `//div[@id='content']/* | //title`.

Pri stránkach Wikipédie je text umiestnený v značke `<div>` s identifikátorom `mw-content-text`. Ako hlavička môže byť použitá značka `<title>`. Preto výsledná podoba vyhľadávacieho reťazca bude `//div[@id='mw-content-text']/* | //title`.

7.2.3 Spracovanie nájdených elementov

Ak sa vyhľadáva `text()`, nájde reťazec, ktorý značka obsahuje. Vyhľadávanie textu nie je vždy vhodné. Na spracovanie je potrebný aj názov značky. Pri niektorých aj ich parametre. Ak sa nevyhľadáva text, ako výsledok bude vrátený objekt `Element`, ktorý bude obsahovať všetky potrebné údaje.

Problém nastáva v prípade, ak je text prvku `None` a pre prvok sa vypisuje text. Pri konvertovaní na Jupyter dokument sa zmení na `null`. Takýto dokument nemôže byť spustený. Texty s hodnotou `None` sú preto nahradené prázdnym reťazcom. Ich prihodením k výslednému textu sa jeho forma nezmení a dokument bude spustiteľný.

Pri nájdení značky sa vytvorí prázdna bunka a pridá sa do dokumentu. Všetky značky v nájdenej značke sú súčasťou novo vytvorenej bunky. Ak by bola výsledná bunka prázdna, do dokumentu nebude pridaná.

Pri značke `<p>` a aj pri iných výnimkách je text vložený bez zvláštnych úprav alebo riadiacich značiek.

Pri značke `<a>` je text vložený do hranatých zátvoriek. Za hranatými zátvorkami je vložený odkaz do normálnych zátvoriek. Odkaz je získaný z atribútu `href` spracovávanej značky `<a>`. `[text](odkaz)`. Problém môže nastať s relatívnymi odkazmi. Jupyter dokument s veľkou pravdepodobnosťou nevytvorí rovnaké záložky ako tie v html dokumente. A ani nemá správne dokumenty na relatívne odkazy. Preto sa relatívne odkazy budú konvertovať na absolútne. Absolútne odkazy sa vytvárajú pomocou kombinácie, zadanej url adresy a relatívneho odkazu. Absolútne odkazy sa vytvárajú iným spôsobom keď je zdroj lokálny súbor namiesto url.

Pri značke `<img>` sa obrázok najskôr stiahne do priečinku s názvom dokumentu. Ak sa konvertuje stránka z internetu, tak sa na stiahnutie použije funkcia `urllib.request.urlretrieve(source, destination)`, pričom zdrojová adresa musí byť absolútna a cieľ musí byť v priečinku z názvom dokumentu. Ak sa konvertuje dokument z disku, tak na

kopírovanie obrázku sa použije funkcia `shutil.copyfile(source, destination)`. Ako načítanie obrázku v markdown sa používa konštrukcia `![text](odkaz na obrázok)`. Ako text sa použije parameter `alt`. Odkazuje sa na stiahnutý obrázok.

Pri značke `<strong>` alebo značke `<b>` je text značky vložený medzi dvojice hviezdičiek. `**text**`. Toto spracovanie je vhodné aj pre značku `<big>`.

Pri značke `<em>` alebo značke `<i>` je text značky vložený medzi dve hviezdičky. `*text*`.

Pri značkách hlavičiek, čo sú značky `<h1>` až `<h6>`, je ich text vložený za znak mriežky. Počet mriežok je určený úrovňou hlavičky. `# text ...###### text`.

Značka `<title>` sa správa ako značka `<h1>`, až na to, že nie je braný do úvahy jej chvost.

Značky `<dd>` a `<dt>` sa riešia ako zjednodušená verzia nezoradeného listu, kde značky `<dd>` budú odsadené o jednu úroveň.

Spracovanie tabuľky sa začína značkou `<table>`. Značka `<tr>` určuje riadok tabuľky a značky `<td>` alebo `<th>` stĺpce tabuľky. Stĺpce v tabuľke markdown sa oddeľujú znakom `|` a riadky sa dávajú na nový riadok. Dôležitý je hlavne špeciálny druhý riadok tabuľky, ktorý určuje štruktúru tabuľky, počet stĺpcov a ich zarovnanie textu. Ak tabuľka v markdown nemusí byť pekná, tak jej tvorba nie je zložitá. Pri značke `<tr>` je potrebné zabezpečiť nový riadok. A pri značkách `<td>` a `<th>` oddelenia znakom `|`. Pri `<th>` je vhodné vložiť aj dvojice hviezdičiek, aby boli zvýraznené. Vloženie špeciálneho druhého riadku je však zložitejšie. Keď sa vkladá, je potrebné vedieť počet stĺpcov a zároveň musí byť druhý v poradí. Počet stĺpcov sa vie v značke `<tr>`. Môže to byť vypísané za prvou značkou `<tr>`. Prvá značka `<tr>` sa však nespoľahlivo zisťuje rekurzívne, preto je na to vhodná globálna premenná. Nepredpokladám tabuľku vnorenú v tabuľke takže nie je potrebný zásobník. Štruktúry, kde sa používa tabuľka vnorená v tabuľke, sú bežnejšie než sa zdá. Ale nepoužívajú sa na zobrazovanie údajov, ale ako riadiaca štruktúra stránky. Riadiaca štruktúra stránky by mala byť vynechaná. Na to slúži prehľadávanie dokumentu a config súbor. Prvá značka `<tr>` sa ťažko zisťuje rekurzívne lebo značka `<table>` nemusí obsahovať iba značky `<tr>` ale bežná je aj značku `<caption>`. Stretol som sa aj tabuľkami, ktoré obsahovali značky `<colgroup>`, `<thead>` a `<tbody>` na stránkach docs.python.org. Ale vytvorený systém funguje aj na ne, lebo obsahujú značky `<tr>`, `<td>` a `<th>`. Značky `<td>` a `<th>` môžu obsahovať aj atribút `colspan`. Pri jeho výskyte sa vloží za bunku daným atribútom udaný počet prázdnych stĺpcov.

Značka `<pre>` označuje už naformátovaný text, na rozdiel od iných značiek ako napríklad `<p>`. Správne formátovanie je pre programový kód dôležité. V niektorých programovacích jazykoch, napríklad Python, je dokonca nevyhnutné na správnu funkciu. Na stránkach, s ktorými som pracoval, sa používa na označenie bloku programového kódu. Ak sa táto značka vyskytuje v bunke, tak sa mení jej typ na `code cell`. Kód je braný implicitne v jazyku Python. K bunke musia byť pridané a nastavené aj ďalšie časti, ako sú napríklad `metadata`.

Značka `<code>` je oveľa špeciálnejšia ako predchádzajúce značky. Označuje časti textu, ktoré majú byť vypísané ako programový kód. Môžu sa nachádzať v texte ako krátky kus kódu v riadku, alebo ako súčasť bloku kódu v značke `<pre>`. Ak sa nachádza ako súčasť textu v markdown bunke, tak musí byť ohradený plotom. `‘text‘`. Ale ak sa jedná o súčasť kódovej bunky, tak musí byť vypísaný bez plotu alebo iných dodatkov.

Medzi fungovaním zoznamov v html a zoznamov v markdown je veľký syntaktický rozdiel. Html používa značky `<ol>` a `<ul>` na určenie typu zoznamu a značku `<li>` na určenie položky zoznamu. Podobný princíp používa aj LaTeX. Markdown pred riadok, ktorý je súčasťou zoznamu, potrebuje špeciálnu sekvenciu: pre číslované zoznamy `1. text` a pre nečíslované `*. text`. Aj vnorenie zoznamov sa rieši rozdielne. U html môže značka zoznamu

`<ol>` alebo `<ul>` obsahovať ďalšiu značku zoznamu, nezáleží akú. U markdown pred sekvenciu na zoznamy môže byť vložené odsadenie medzerami. Dve medzery spravia jednu úroveň vnorenia. `*. text`. Po zvážení daných okolností som sa rozhodol spraviť na zoznamy zásobník. Keď príde značka `<ul>` vložím do zásobníku reťazec `unordered`. A keď príde značka `<ol>` vložím do zásobníku reťazec `ordered`. Keď príde na rad značka `<li>`, tak typ zoznamu sa dá určiť podľa reťazca na vrchole zásobníku. Ak je na vrchole zásobníku reťazec `unordered`, tak pred text, ktorý obsahuje značka `<li>`, pridám špeciálnu sekvenciu `*.` a za text je ešte potrebné pridať nový riadok `*. text \n`. Ak je na vrchole zásobníku reťazec `ordered`, tak musím získať poradové číslo prvku daného zoznamu. V html kóde sa nenachádza. Značky spracovávam rekurzívnou funkciou, preto zriadiť na to globálnu premennú je nepostačujúce. Pri volaní rekurzívnej funkcie som umožnil zadať jej poradové číslo. Stačí len čítať poradie značky pri spracovaní. Dá sa na to použiť indexovacia premenná alebo funkcia `range(start, stop)`. Ale ako viac pythonovský prístup som považoval funkciu `enumerate(iterable, start=0)` zo štartom jedna. Potom už zostrojenie špeciálnej sekvencie je jednoduché. `1. text \n`. Po dokončení spracovania prvkov zoznamu je potrebné odstrániť reťazec na určenie typu zoznamu z vrchola zásobníku. Zoznamy je potrebné ešte správne odsadiť. Na správne odsadenie zoznamov je potrebné vedieť či sa zoznam nenachádza v ďalšom zozname alebo v ďalších zoznamoch. To bol jeden z dôvodov použitia zásobníka namiesto jednej premennej. Podľa počtu prvkov v zásobníku sa dá určiť úroveň vnorenia rozoberaného zoznamu. Pred prvok zoznamu treba preto vložiť určený počet medzier $2 * (\text{počet prvkov v zásobníku} - 1)$. $2*$ je jednoducho vyriešené vložením dvoch medzier naraz. Ale nie pre každý prvok, jeden musí byť vynechaný. `zasobnik[1:]` vynechá prvý prvok. Takže cyklus používajúci zasobnik[1:] sa vykoná počtom prvkov v zásobníku bez jedného.

Ďalší problém so spracovaním nastáva s markdown funkcionalitou, ktorá potrebuje ukončovací znak alebo reťazec. Napríklad `[text](odkaz)` alebo `**text**`. Text nieje len text značky. Zahŕňa aj text všetkých značiek, ktoré rozoberaná značka v sebe obsahuje. A to ide hlbšie, až kým sa nespracujú všetky značky, ktoré rozoberaná značka obsahuje. V tom je ďalšia výhoda použitia rekurzívnej funkcie. Môže byť vytvorená lokálna premenná, ktorá je nezmenená rekurzívnymi volaniami. Pri spracovaní značky sa tam ukončovací reťazec alebo znak uložia. Po spracovaní všetkých značiek, ktoré rozoberaná značka obsahuje, môže byť vypísaná. Bez tejto úpravy by html kód `<a href="https://tools.ietf.org/html/rfc7159.html"><strong>RFC 7159</strong></a>` bol konvertovaný do nesprávneho tvaru `[](https://tools.ietf.org/html/rfc7159.html)**RFC 7159**` namiesto správneho tvaru `[**RFC 7159**](https://tools.ietf.org/html/rfc7159.html)`.

Pri markdown bunkách je potrebné ošetriť znaky _ a *. Sú to riadiace znaky systému markdown. Ich riadiaca funkcia sa dá anulovať znakom \. Znak \ ruší riadiacu funkciu znaku idúceho za ním. Tieto znaky nemajú riadiaci charakter v bunkách typu code a ani, keď sú oplotené (`‘text‘`). A to platí aj pre všetky značky, ktoré sú v nich. To sa dá najlepšie riešiť rekurzívne. Značky sú natívne v móde výpisu markdown. Svoj mód výpisu odovzdávajú aj značkám, ktoré obsahujú. Ak sa dorazí na značku `<code>`, tak zmení mód výpisu na typ `code`. To isté platí aj pre značku `<pre>`. V móde výpisu `code` neplatia ani iné riadiace prvky systému markdown. Aj keď sa v značke `<pre>` môže vyskytovať odkaz, tak v kódovej bunke sa nachádzať nemôže. Preto nie je vhodné vypisovať riadiace značky systému markdown v kódovej bunke. Výnimkou sú aj jednoriadkové príkazy systému markdown. Napríklad zoznamy a tabuľky. Nový riadok má pri nich dôležitý význam, to pri html neplatí. V html sa formátovanie pomocou bielych značiek počíta iba pri značke `<pre>`. V ostatných prípadoch sa biele značky zúžia na jednu medzeru.

Značky sa môžu vyskytovať aj uprostred textu. Napríklad značka `<a>` uprostred paragrafu `<p>`. Ak sa chvost značky `<a>` nevypíše, tak zvyšok paragrafu `<p>`, nasledujúci za značkou, bude vynechaný. Preto chvost značky bude vrátený značke, ktorá zavolala spracovanie značky. Ak je text značky použitý, tak po spracovaní značky, ktorú obsahuje, vypíše vrátený chvost značky. Text značky nie je potrebný pri značkách ako `<ol>` alebo `<ul>`.

7.3 Nepriame rozšírenie nbextension

Rozšírenie nemôže byť vyrobené čisto v javascripte, ale nie je to nevyhnutné. Jupyter dokument môže spúšťať aj kód v jazyku Python. Kód na získanie, izolovanie a konvertovanie dokumentu máme už z predchádzajúceho systému. Treba ho len upraviť na fungovanie v Jupyter dokumente.

7.3.1 Generovanie buniek

Výsledok konverzie je vloženie buniek so správnym obsahom do dokumentu, v ktorom bola konverzia spustená.

V Python jadre pre Jupyter dokument je zabudovaná funkcia `get_ipython().set_next_input(text)`. Vytvorí bunku typu `code` s textom špecifikovaným v argumente funkcie. Lenže aj pri viacerých volaní funkcie vytvorí iba jednu bunku s textom posledného zavolania funkcie. Predpokladám, že bunka sa vytvára až po skončení vykonávania kódu. A pri použití funkcie `get_ipython().set_next_input(text)` sa prepíše text bunky, ktorá bude vytvorená.

Keďže je potrebné vytvoriť viacero buniek, funkcia `get_ipython().set_next_input(text)` nepostačuje. Python jadro pre Jupyter dokument má aj funkciu `IPython.display.display_javascript()`, ktorá dokáže spustiť kód v javascripte. V javascripte sa bunka dá vytvoriť použitím funkcie `IPython.notebook.insert_cell_below()`, ktorá vráti objekt bunky. Nad vráteným objektom bunky môže byť použitá funkcia `set_text(text)`, ktorá nastaví text novo vytvorenej bunky. Ďalšie funkcie na bunke sa nedajú spustiť nad objektom bunky. Preto je potrebné získať index bunky, ktorú sme vytvorili. Dá sa získať funkciou `IPython.notebook.get_cells().indexOf(cell)`. Použitím indexu bunky sa dá konvertovať na markdown, alebo code. Bunka sa dá vybrať alebo sa dá na ňu zamerať. Na to, aby sa bunky vykladali v správnom poradí, musí sa výber bunky posunúť na vytvorenú bunku. Ak by sa výber neposunul, tak by boli bunky po konverzií v opačnom poradí. Funkcia `IPython.notebook.to_markdown(index)` výber posunie. Ale funkcia `IPython.notebook.to_code(index)` výber neposunie. Preto pri vytváraní kódovej bunky sa použije aj funkcia `IPython.notebook.select(index)`.

7.3.2 Spustenie programu

Keďže je program v jazyku Python, spúšťa sa spustením bunky. Kód sa dá do bunky vložiť manuálne. Ale ako lepšie riešenie je automatické generovanie bunky. Po stlačení tlačidla môže byť vytvorená nová bunka s programom v nej a nastaveniami na začiatku programu. Generovaný Python kód môže byť uložený v premennej javascriptu. Ak je v premennej, ťažko sa modifikuje a nezvýrazňuje sa syntax. Nie je to problém pre krátke kódy, ale pre dlhšie áno. Bolo by vhodné mať kód konvertora v externom súbore a do javascriptu ho načítať pri vytvorení bunky. Javascript z bezpečnostných dôvodov nevie voľne

pracovať s lokálnymi súbormi. Ale pomocou kombinácie funkcií z `jquery` a `require`. Zápisom `\$.get(require.toUrl("./file.py"), function(file_text))` sa dá získať text súboru, ktorý je súčasťou rozšírenia. Týmto spôsobom môže byť text súboru vložený do generovanej bunky.

Bunka s celým kódom programu na konvertovanie je v celku dlhá. Bolo by ju dobré skrátiť. Z javascriptu sa dá pomocou funkcie `IPython.notebook.kernel.execute(python _code)` spustiť kód, bez toho aby bol v bunke. Konvertor som prerobil na triedu. Trieda je potom definovaná pomocou príkazu `IPython.notebook.kernel.execute(python_code)` a môže byť použitá v ďalších bunkách. Stačí na konvertovanie stránky vygenerovať len bunku s nastavením a spustením konvertora. Týmto sa zníži dĺžka bunky. Inštancia konvertora je vytvorená príkazom `insert_html_page(url, xpath, image_dir)`. Kde `url` je odkaz na stránku. Môže byť stránka na internete začínajúca na `http://` alebo `https://`. `xpath` je izolačný reťazec vo formáte xpath, ktorý záleží na konvertovanej stránke. A `image_dir` je adresár, do ktorého budú uložené obrázky patriace k stránke. Konverzia sa spustí vyvolaním funkcie `start()` nad inštanciou konvertora. Ako súčasť optimalizácie je dobré, ak sa trieda konvertora definuje iba raz. To bude docielené vnútornou premennou rozšírenia. Pri načítaní rozšírenia bude nastavená na `true`. Pri vytvorení bunky na spustenie konvertora, ak je daná premenná nastavená na `true`, tak sa spustí definícia konvertora a premennej sa zmení stav na `false`.

7.3.3 Konfigurácia

Je potrebné upraviť konfiguráciu rozšírenia. Konfigurácia sa dá vyrobiť staticky, ale bolo by vhodné, ak užívateľ môže ku konfigurácií pridať vlastné záznamy bez úpravy kódu. Nbextensions sa dajú bežne konfigurovať cez Jupyter Nbextensions Configurator. Parametre, ktoré sa dajú nastavovať, sa určujú v yaml súbore rozšírenia. Na vytvorenie konfigurácie je potrebné mapovanie domény na izolačný reťazec. Na to je vhodný typ parametru `list`. Zoznam bude obsahovať textové položky. Formát položky musí byť: „doména = izolačný reťazec“. Týmto spôsobom užívateľ môže s Jupyter Nbextensions Configurator pridať vlastné nastavenia a upraviť alebo odstrániť základné nastavenia.

Izolačné reťazce sa v paremetroch pre rozšírenie definujú ako zoznam, kde element, ktorý uchováva izolačný reťazec, je definovaný ako text. V yaml súbore musia byť základné nastavenia. Na sprístupnenie konfigurácie v rozšírení sú potrebné knižnice `base/js/utils` a `services/config`. Konfigurácia sa nastaví použitím funkcií `config.ConfigSection( 'notebook', {base_url: utils.get_body_data("baseUrl")})`. `config` a `utils` reprezentujú potrebné knižnice. Konfigurácia sa načíta funkciou `config.load()`. Reakcia na načítanie konfigurácie sa nastaví funkciou `config.loaded.then()`. Už načítané parametre konfigurácie sú v premennej `config.data`. Základné nastavenia musia byť aj v javascripte. Ak konfigurácia bola nezmenená, tak `config.data` budú prázdne. Pri spracovaní konfigurácie sa musí overiť existencia dát. Ak neexistujú, tak sa použijú základné. Ak existujú, tak sa text rozdelí na doménu a izolačný reťazec.

K nastaveniu nbextensions sa dá pristupovať len v javascripte. Už pri generovaní kódovej bunky treba vedieť zdroj, aby mohol byť vybraný správny reťazec z konfigurácie. Preto pri generovaní bunky sa vyvolá funkcia `prompt()`, v ktorej by mal užívateľ zadať url. Ak bola url adresa zadaná, tak sa z konfigurácie vyberie izolačný reťazec. Podľa plnej domény alebo jej skrátenej verzie. Týmto spôsobom užívateľ môže izolačný reťazec pred spustením konverzie zmeniť a prispôsobiť tým konverziu pre špeciálne prípady.

K nastaveniam patrí aj výber priečinku, do ktorého budú uložené obrázky patriace ku konvertovanému dokumentu. Predvolený názov priečinku je názov dokumentu. Názov dokumentu sa dá odvodiť z cesty dokumentu, ktorá môže byť získaná z `IPython.notebook .notebook_path`.

7.3.4 Vstup pomocou schránky clipboard

Narozdiel od oddeleného skriptu v jazyku Python, rozšírenie vstavané v systéme Jupyter môže byť použité na vloženie kratších častí stránky. Na vloženie kratších častí stránky môže byť použitý upravený izolačný reťazec, aby mohol užívateľ odkázať na správny element pomocou izolačného reťazca. Užívateľ musí poznať jazyk html, konštrukciu stránky a systém xpath. Na zistenie konštrukcie stránky sa môže použiť prehľadávací nástroj vstavaný v prehliadači. Najjednoduchšie sa dá otvoriť vyvolaním kontextového menu nad miestom, ktoré chce užívateľ vložiť do dokumentu a vybraním možnosti preskúmať prvok. Jednoduchšie by bolo použiť schránku clipboard. Predovšetkým pre bežného používateľa. Do schránky clipboard sa vkladá označením elementov a skratky `ctrl+c`. Alebo vyvolaním kontextového menu a vybraním možnosti kopírovať. Schránka clipboard ukladá dáta vo viacerých formátoch. Aj vo formáte html, ktorý sa používa pri konštrukcií stránky a aj ako vstup konvertovania. Ak užívateľ skopíruje údaje, ktoré potrebuje vložiť, potom užívateľ musí byt schopný povedať konvertoru, aby bral do úvahy dáta zo schránky clipboard namiesto sťahovania stránky z internetu alebo otvárania lokálneho súboru. Zdroj dát sa udáva pomocou reťazca, ktorý môže byť url alebo cesta k súboru. Môžu byť vyhradené kľúčové slová, ktoré určia zdroj ako schránku clipboard. Ako jedno kľúčové slovo môže byť `clipboard`. Ďalší špeciálny prípad môže byť prázdny reťazec. Nie je to správne url na nájdenie html stránky. Prázdny reťazec nie je správna cesta k súboru, ale kľúčové slovo `clipboard` môže byť názov súboru.

Na prácu so schránkou clipboard sa používa knižnica `win32clipboard`. Najskôr musí byť schránka otvorená použitím funkcie `win32clipboard.OpenClipboard(0)`. A po získaní údajov sa zatvorí použitím funkcie `win32clipboard.CloseClipboard()`. Dáta zo schránky sa získajú funkciou `win32clipboard.GetClipboardData(format)`. Kde `format` je číslo reprezentujúce formát dát, o ktoré sa žiada. Číslo formátu sa získa použitím funkcie `win32clipboard.RegisterClipboardFormat(text_format)`. Kde `text_format` je textová reprezentácia formátu. Html formát je reprezentovaný reťazcom `"HTML Format"`. Získané html dáta musia byť najskôr dekódované z binárnych dát na textový reťazec. Dekódujú sa použitím funkcie `.decode("utf-8")`. Získané dáta v textovom formáte neobsahujú iba skopírovanú časť html stránky ale aj meta-dáta. Obsahuje údaje o fragmente a sekcii. Ak boli skopírované z html stránky môže obsahovať aj adresu stránky, z ktorej boli skopírované. Adresa stránky musí byť použitá na zostavenie relatívnych odkazov. Je označená kľúčovým slovom `SourceURL:` a môže byť získaná regulárnym výrazom. Html fragment je zabalený v značkách `<html>` a `<body>`, aj keď ich neobsahoval. Ak schránka nemá html dáta, tak na konvertovanie môže byť použitý čistý text. Získanie html dát musí byť v sekcii zachytenia chyby `try:`. Ak html dáta neobsahuje, vyvolá chybu `TypeError`. Čistý text sa dá získať zo schránky bez udania formátu príkazom `win32clipboard.GetClipboardData()`. Aj bez formátu môže nastať chyba `TypeError`, ak je schránka prázdna [20].

Kapitola 8

Obrázky vstavané v dokumente

Jupyter je postavený na html základe. Html nepodporuje obrázky v dokumente. Obrázky musia byť v externých súboroch a sú v html odkazované. Tento systém funguje pre web stránky. Ale nefunguje pre systémy, ktoré majú byť prenositeľné. K Jupyter dokumentu, ktorý používa obrázky, treba distribuovať aj obrázky. Takže je potrebné distribuovať viacero súborov alebo balíček. Tento problém rieši mnou vytvorené rozšírenie, ktoré umožní ukladať obrázky do meta-dát dokumentu.

8.1 Base64

Base64 je systém kódovania textu aj obrázkov do jednoduchého textu. Môže byť použitý v html ako zdroj obrázka. Zdroj obrázka v base64 má formu: `data:image/%image_format%;[charset=utf-8;]base64, %base64_data%`. `%image_format%` je formát obrázku, napríklad `jpg` alebo `png`. Položka znakový set(`charset`) je voliteľná. `%base64_data%` je dlhý reťazec v kódovaní base64, ktorý reprezentuje obrázok.

8.1.1 Použitie base64 v Jupyter dokumente

V Jupyter dokumente môžeme obrázok zobraziť napríklad v markdown bunke. Použitím zápisu `![Alt text](source)`. Takýto zápis má výhodu, že môže byť v texte a jeho zápis nie je viditeľní po spustení bunky. Zdroj môže mať base64 formu, ale pri spracovaní markdown bunky je base64 reťazec odstránený. V markdown môžu byť aj html značky. Napríklad aj značka `<img>`, ktorej zdroj môže byť v base64. Ale aj tento zápis má ten istý problém že base64 reťaz bude pri spracovaní odstránený.

Obrázky sa dajú vložiť aj cez bunku s kódom. Použitím čarovného príkazu `%%html`, kód bunky sa bude považovať za html kód. Môže byť použitá značka `<img>` s base64 zdrojom. Pri spustení kódovej bunky sa base64 reťaz neodstráni.

Použitím čarovného príkazu `%%javascript` kód bunky sa bude považovať za javascript kód. V ňom sa dá vytvoriť obrázok, ale javascript nemá výstup pre bunku. Preto sa obrázok nedá jednoducho zobraziť.

Obrázok sa dá vložiť aj použitím jazyku Python. V knižnici `IPython` je funkcia `display.Image(source)`, ktorá zobrazí obrázky. Zdroj môže byť cesta k obrázku, ale aj base64 dáta. Pre túto funkciu sa nepoužíva celý base64 reťazec. Používajú sa iba base64 dáta bez formátovania. Funkcia `Image()` si formátovanie doplní.

Pri spracovaní markdown bunky sa base64 reťazec odstráni. Ale k obsahu markdown bunky sa dá dostať po konvertovaní. Rozšírenia, ktoré modifikujú výpis markdown bunky,

pristupujú k bunke až po jej prvotnom spracovaní. Rozšírenie môže zachytiť v bunke špeciálny identifikátor a na jeho miesto vložiť base64 obrázok v html značke.

8.2 Vloženie obrázka do dokumentu

Obrázky v markdown bunke, ktoré používajú kódovanie base64, budú reprezentované obrázkom zo zdrojovou adresou začínajúcou na `image.base64/`. Namiesto týchto zástupných obrázkov bude rozšírením vložené na ich miesto obrázok, na ktorý sa odkazujú. Base64 dáta, ktoré sú dlhé, nemusia byť uložené v markdown bunke. Zlepší sa tým jej prehľadnosť. Base64 reťazce sú uložené v meta-dátach dokumentu. Obrázok stačí mať v meta-dátach raz, ale môže byť odkazovaný na viacerých miestach.

Spracovanie markdown bunky sa deteguje udalosťou `rendered.MarkdownCell`. Obrázky sa v markdown bunke nájdu pomocou regulárneho výrazu `/<img .*?src="(.*?)".*?\/?> /g`. Regulárny výraz je vyvolaný funkciou `replace()`. Nahradenie je vykonávané cez funkciu. Zavolá sa keď funkcia `replace()` nájde niečo na nahradenie. Ak sa zdroj začína na `image.base64/`, tak sa ho pokúsi nájsť v meta-dátach dokumentu. Zdroj obrázka je nahradený nájdeným base64 reťazcom. Ak sa zdroj nedá nájsť v meta-dátach dokumentu, tak sa vypíše hláška do konzoly a zdroj zostane nezmenený. Ak užívateľ nemá toto rozšírenie, tak sa upravenie markdown bunky neuskutoční. Táto vlastnosť môže spôsobiť problémy z prenositeľnosťou dokumentu.

V dialógovom okne obrázka je zoznam všetkých importovaných obrázkov. K obrázku je pridané tlačidlo, ktoré môže vygenerovať kódovú bunku typu html. Bude vygenerovaná pod vybranú bunku. Ako prvé musí obsahovať čarovný príkaz `%%html`. Potom bude obsahovať element `<img>`. Jeho atribút `alt` je názov obrázka. A atribút `src` obsahuje reťazec vo formáte base64, ktorý patrí k danému obrázku. Po spustení kódovej bunky v jej výstupe bude daný obrázok. Zobrazenie pomocou kódovej bunky typu html funguje aj bez rozšírenia, pretože base64 reťazec je súčasťou html bunky. Ale na rozdiel od predchádzajúceho riešenia sa obrázok automaticky neaktualizuje.

8.3 Importovanie obrázka

Javascript dokáže konvertovať obrázky do formátu base64. Do base64 sa môžu konvertovať obrázky vo formáte png, jpg a gif. Pre konvertovanie je formát získaný z prípony obrázka. Obrázok musí byť najskôr načítaný v objekte `Image`. Načítanie obrázka je zachytené udalosťou `load` nad vytvoreným objektom. Obrázok musí byť možné načítať v html elemente `<img>`, aby ho bolo schopné previesť na base64 reťazec. Base64 reťazec sa môže získať vykreslením načítaného obrázka v elemente `<canvas>`. Najskôr sa elementu `<canvas>` nastaví šírka a výška podľa načítaného obrázka. Získa sa jeho $2D$ kontext, v ktorom bude načítaný obrázok vykreslený na počiatočnej pozícií(0,0). Base64 reťazec sa z elementu `<canvas>` získa funkciou `toDataURL()`.

Konvertovať obrázok do base64 sa dá aj pomocou objektu `FileReader`. Dokáže asynchrónne čítať obsah súborov, ktoré sú načítané z počítača používateľa. Súbor musí byť získaný z elementu `<input>`. Používateľ ho musí vybrať použitím správcu súborov. Javascript nedokáže pracovať priamo so súborovým systémom z bezpečnostných dôvodov. Funkcia nad čítačom `reader.readAsDataURL(file)` prečíta súbor a skonvertuje ho na base64 reťazec. Výsledok konverzie sa uloží v premennej objektu `reader.result`. Výsledok je platný až

po načítaní súboru. Úspešné načítanie sa zachytí pri udalosti `load` nad objektom `reader`. Neúspešné načítanie sa zachytí pri udalosti `error` nad objektom `reader`.

Získaný base64 reťazec bude vložený do meta-dát dokumentu. Ak je vstup adresa obrázku, tak položka, na ktorú bude base64 reťazec vložený, je odvodená z cesty obrázka s predponou `image.base64/`. Ak je vstup súbor, tak je položka odvodená z názvu súboru s predponou `image.base64/`. Kvôli bezpečnosti cesta k súboru nie je v javascripte dostupná.

8.4 Dialóg

Systém na vkladanie obrázkov do meta-dát dokumentu treba užívateľovi sprístupniť. Vstup systému na vkladanie obrázkov je cesta k obrázku. Výstup je načítanie obrázka a upravenie meta-dát dokumentu. Alebo aj neschopnosť načítať a konvertovať obrázok.

K pridaniu obrázkov musí patriť aj zmazanie obrázkov. Ako vstup je názov položky v meta-dátach, ktorá reprezentuje obrázok. Ako výstup je upravenie meta-dát. Ak sa položka, ktorá reprezentuje obrázok, v meta-dátach nenachádza, tak sa to môže považovať za úspešné vymazanie.

Užívateľ by mal byť schopný pozrieť, aké obrázky už má načítané, a aj vedieť získať base64 reťazec, na ktorý boli skonvertované.

Na sprostredkovanie týchto vlastností môžu byť použité funkcie `prompt()` a `alert()`. Funkciou `prompt()` sa dá získať od užívateľa textový vstup a funkciou `alert()` sa dajú zobraziť správy a údaje. Ale systém na spravovanie obrázkov je v niektorých úkonoch príliš zložitý. Vyžadoval by viacero použití dialógov `prompt()` a `alert()` na jeden úkon. Systém na spravovanie obrázkov sa dá vyrobiť použitím funkcií `prompt()` a `alert()`, ale budú sa najskôr hľadať lepšie možnosti komunikácie s užívateľom.

Ako ďalší nástroj na komunikáciu s užívateľom je možné použiť jquery dialóg. Knižnica jquery sa bežne používa v systéme Jupyter. Už je zahrnutá v systéme pre iné funkcie a netreba ju načítavať len kvôli spravovaniu obrázkov. Volá sa funkciou `$(element).dialog(nastavenia)`, kde element je základný prvok, ktorý bude vybraný z dokumentu a zobrazený. Okolo tohoto elementu sa budú odohrávať funkcie dialógu. Element môže mať html, css aj javascript časti. Pri písaní javascript kódu pre element treba dávať pozor na to, že vytvorený element nie je v rovnakom kontexte ako kód, ktorý ho vytvára. Funkcie deklarované vo vnútri rozšírenia sa nedajú použiť mimo rozšírenia. Problém môže nastať pri vytváraní dynamických reakcií na prvky elementu. Tento problém sa ale dá obísť vytváraním prvkov elementu, ako objekty použitím funkcie `document.createElement(name)`. Takto vytvorené objekty sa pridajú k elementu funkciou `element.appendChild(prvok)`. Prvky sa vytvárajú upravením html kódu vnútorného elementu funkciou `element.innerHTML = prvok`. Všetky funkcie musia byť priamo v prvku. Táto limitácia robí problémy pri zložitejších vlastnostiach systému. Nastavenia pre jquery dialóg sú reprezentované objektom. Objekt obsahuje prvky, ktoré upravia špecifické správanie dialógu. Dokážu upraviť výzor dialógu, nastaviť reakcie na špeciálne udalosti dialógu a pridať aj prvky mimo hlavného elementu, napríklad tlačidlá. Ak prvky nebudú nastavené, tak budú mať natívnu funkciu. Nie všetky nastavenia budú zaujímave na spravovanie obrázkov. Zaujímavé nastavenia výzoru dialógu pre spravovanie obrázkov sú šírka a poloha. Natívna šírka nie je dosť veľká na zobrazenie všetkých potrebných údajov a funkčných prvkov. Natívna poloha jquery dialógu je stred stránky. Pre potreby práce Jupyter dokumentom je lepšie posunúť dialóg vyššie. Pozícia sa dá nastaviť aby bola pod panelom nástrojov. Na potreby spravovania obrázkov je zaujímavá udalosť zatvorenia a otvorenia dialógu.

Keď sa jquery dialóg zatvorí, vzniká po ňom mátoha. Ak by tam tá mátoha nebola, tak sa môžu elementy dialógu odkazovať pomocou ich identifikátorov bez toho, aby vzniklo viacero nálezov, ktoré sú v mátohe. Bez nálezov v mátohe sa môže s týmito elementami pracovať a vieme, že budú vždy v aktívnom okne. Ak sa mátoha neodstráni manuálne, tak v dokumente ostane, len že sa nebude zobrazovať `display: none`. Ak sa ako jquery selektor použije identifikátor, tak sa očakáva len jeden výsledok a ďalšie budú ignorované. Preto funkcie, ktoré aktualizujú dialógové okno, môžu pracovať s mátohou namiesto aktívneho dialógového okna. Ako jquery selektor môže byť použitá trieda. V takomto prípade sa očakáva viacero výsledkov. Takto funkcie, ktoré aktualizujú dialógové okno, budú pracovať s aktívnym oknom aj s každou mátohou. Môže to zbytočne zdržovať vykonávanie. Práve pri udalosti zatvorenia sa dá mátoha zrušiť. Zrušenie mátohy ušetrí aj priestor v pamäti. Ďalší spôsob ako obísť mátohy je im poslať odkaz dialógového okna, z ktorého boli zavolané. Odkaz na prvok, ktorý zavolal funkciu, je v premennej `this`. Pri udalosti to je element, nad ktorým je udalosť registrovaná, v našom prípade dialógové okno, a pri kliknutí to je tlačidlo. Táto situácia sa dá zmeniť použitím funkcie `bind()`. Funkcia `bind()` vloží do inej funkcie premenné. Zmení premennú `this` a premenné pred ostatnými. Týmto spôsobom sa všetkým funkciám pracujúcim s oknom môže explicitne predať okno dialógu a aj iné premenné, ktoré sú potrebné pre ich funkciu a časom sa menia. Vďaka tejto úprave sa pri všetkých funkciách, ktoré pracujú s oknom, môže použiť konštrukcia jquery `$(this).find(selektor)`. Táto konštrukcia narozdiel od konštrukcie `$(selektor)` nájde elementy podľa jquery selektoru ale iba v dialógovom okne, z ktorého boli zavolané. Ďalším spôsobom ako obísť mátohy je použiť odkazy na susedné elementy. Pri tlačidle sa dá získať vstup cez `this.previousElementSibling`, ak je element `<input>` pred tlačidlom. Pri udalostiach sa dajú získať elementy dialógového okna pomocou `this.children` alebo aj `$(this).find(selektor)`. Dialógové okno sa dá z elementu v ňom získať cez vlastnosť `this.offsetParent`. Ukazuje na najbližší rodičovský element z pohľadu štruktúry DOM stromu, ktorý má pozíciu relatívnu `position: relative` alebo absolútnu `position: absolute`. Absolútna alebo relatívna pozícia spôsobí, že element je mimo normálnej konštrukcie stránky. To pre jquery dialóg platí. Ale nie je to veľmi spoľahlivé. Pre `offsetParent` platí viacero výnimiek, ktoré môžu spôsobiť odkaz na zlý element.

8.4.1 Vloženie obrázka

Ako prvé treba pripraviť systém, s ktorým môže užívateľ požiadať o vloženie obrázka. K vloženiu obrázka je potrebná cesta k nemu vo forme reťazca, alebo súbor. Na cestu k obrázku je vhodný element `<input>` typu `text`. Na vstupný súbor je vhodný element `<input>` typu `file`, ktorý akceptuje obrázky. Na spustenie procesu vloženia obrázka je vhodné tlačidlo. Oba vstupné systémy by mali mať vlastné tlačidlá, pretože pracujú oddelene. Spustenie konverzie je tiež vhodné pri stlačení `Enter` na textovom reťazci, ktorý sa používa na zadanie cesty pre vloženie obrázka. Element `<input>` typu `file` má na identifikáciu nastavený identifikátor na `image_file_path_f`. Vstup zo súboru natívne prijme iba jeden súbor, ale sa dá nastaviť aby prijal viacero súborov. Nastavuje sa buď skráteným zápisom `<input type="file" name="img" multiple>`. Slovo `multiple` spôsobí nastavenie. Alebo rozšírený zápis `<input type="file" name="img" multiple="multiple"`. Atribút `multiple="multiple"` spôsobí nastavenie [13]. Použití štýl vytvárania elementov znemožňuje použiť skrátený zápis, preto bol použitý rozšírený. Vybrané súbory sa nachádzajú vo vlastnosti `files` elementu `<input>`. Pri konverzii sa získajú použitím jquery zápisu `$(this).find('#imag_file_path_f').prop('files')`. `this` je odkaz na dialógové

okno. `find('#imag_file_path_f')` nájde v dialógovom okne element s identifikátorom `image_file_path_f`, ktorý reprezentuje vstup pre súbory. A `prop('files')` získa vlastnosť `files`, v ktorej sú načítané súbory. Pre každý súbor, ktorý je zadaný, sa vytvorí nový čítač súborov `FileReader`. Pri konverzií sa premenné musia predať funkciám, ktoré sú priradené udalostiam `load` a `error`, použitím funkcie `bind()`. Ak sa použijú ich globálne varianty, tak pri vyvolaní udalostí `load` a `error` už nemusia byť aktuálne. Funkciám je predaný čítač, ktorý obsahuje výsledok konverzie. Objekt súboru, ktorý obsahuje názov súboru. Názov súboru je použitý pre vytvorenie záznamu do meta-dát dokumentu. A odkaz na dialógové okno, ktorý je použitý na vyvolanie udalostí nad dialógovým oknom.

Stav konvertovania obrázka by mal byť viditeľní užívateľovi. Začatie konverzie a predovšetkým jej úspešnosť alebo neúspešnosť. Indikátor stavu konverzie by mal byť viditeľný bežnému užívateľovi. Preto konzola nepostačí. Mal by byť spojený s prvkami, ktoré sú potrebné na konvertovanie, so správnym vstupom a tlačidlom. Jednoduchá možnosť je k prvkom na konvertovanie pridať textový prvok, ktorý bude ukazovať na stav konverzie. Stavy, ktoré by mohol ukazovať, sú: „začatie konverzie“, „konverzia úspešná“, „konverzia neúspešná“. Elegantnejšie by mohlo byť použitie farieb. Oranžová môže značiť začiatok konverzie. Červená môže značiť, ak bola konverzia neúspešná. Zelená môže značiť, ak bola konverzia úspešná. K prvkom na konvertovanie môže byť priradený nový prvok. Indikátor môže byť kruh, ktorý má reprezentovať kontrolné svetlo. Geometrické tvary sa dajú vytvoriť pomocou elementu `<canvas>`. Ale na vytvorenie kruhu je jednoduchšia metóda. Môže byť použitý element `<div>`. Musí mať pevnú a rovnakú šírku `width` aj výšku `height`. To vytvorí štvorec $width = height = x$. Mal by mať nastavenú hranicu `border`. A radius hranice `border-radius` musí byť nastavený na polovicu hrany štvorca. Polovicu výšky alebo šírky $border - radius = \frac{x}{2}$ [16]. To vytvorí kružnicu. Kružnica sa dá vyplniť nastavením farby pozadia `background-color`. Netreba vytvárať nový element. Zmeny sa môžu prejavovať aj v už existujúcich prvkoch. Buď v stupe alebo v tlačidle. Na tlačidlo môže byť použitá ikona. Pretože tlačidlo slúži na pridanie, jeho ikona bude znak plus. Jupyter používa na ikony Font Awesome. Má dve rôzne ikony na plus. Samotné plus alebo plus v krúžku. Namiesto farby externého indikátoru sa môže meniť farba ikony v tlačidle. Farba ikony reaguje na css atribút `color`. Plus v krúžku zaberá viacej priestoru, preto zmena farby môže byť výraznejšia. Zmena farby bude urobená ako udalosti. Budú rozdelené na dve udalosti definované nad dialógovým oknom. Udalosť nad textovým vstupom bude pomenovaná `status_change`. Udalosť nad súborovým vstupom bude pomenovaná `status_f_change`. Udalosti budú registrované pri otvorení dialógového okna. Registrácia prebehne príkazom `$(this).bind('status_change', function)`. Pri spustení udalosti sa vykoná priradená funkcia. Nájde element, ktorého farbu treba zmeniť podľa jeho identifikátora. Atribút css je nastavený jquery kombináciou `$(this).find('#identifikátor').css('atribút', farba)`. Pri udalosti `status_change` s textovým vstupom je identifikátor pomenovaný `status`. Atribút je farba pozadia `background-color`. A pri udalosti `status_f_change` so súborovým stupom je identifikátor pomenovaný `status_f`. Atribút je farba ikony `color`. Farba je zadaná argumentom udalosti. Udalosť sa spúšťa použitím jquery kombinácie `$(this).trigger('status_change', farba)`. `this` je odkaz na dialógové okno, a `'status_change'` je názov udalosti, ktorý môže byť aj `'status_f_change'`. Názov udalosti záleží od kontextu konverzie. Farba je reťazec, ktorý reprezentuje farbu v css zápise. Farba odovzdaná udalosti záleží od stavu konverzie. Na začiatku konverzie bude zavolaná správna udalosť s oranžovou farbou `$(this).trigger('status_change', 'orange')`. Ak bola konverzia úspešná a nové údaje budú zapísané do meta-dát, tak sa farba zmení na zelenú zavolaním udalosti príkazom `$(dialog_window).trigger('status_change', 'green')`. Ak

obrázok nemôže byť načítaný, tak sa farba zmení na červenú zavolaním udalosti príkazom `$(dialog_window).trigger('status_change', 'red')`. Nenačítaný obrázok môže byť spôsobený prázdnym vstupom. Tak isto aj chybou pri načítaní alebo neexistujúcim vstupom. Pri vstupe viacerých súborov nie všetky súbory môžu dosiahnuť rovnaký stav. Pri textovom riešení môže byť spočítané, koľko konverzií uspelo, koľko obrázkov nebolo možné skonvertovať a ktoré to boli. Pri farebnom riešení je to už väčší problém. Ale môže byť použitá kombinácia týchto systémov. Pod prvkami na vstup pre súbory môže byť umiestnený textový prvok. Bude obsahovať zoznam obrázkov, ktoré boli zadané, a k nim patriace stavy. Obrázok bude identifikovaný jeho menom. Stavy môžu byť zobrazené ako farebné kruhy. Pri pustení konverzie sa zobrazovací prvok vynuluje do základného stavu. Pri začatí konverzie obrázka sa vloží do zobrazovacieho prvku záznam o ňom a indikátor stavu bude nastavený na oranžovú. Ak konverzia prebehla úspešne, tak indikátor stavu bude nastavený na zelenú. A ak konverzia prebehla neúspešne, tak indikátor stavu bude nastavený na červenú. Keďže tento systém je komplikovanejší než konvertovanie iba jedného súboru, je potrebné rozšíriť argument predávaný udalosti. K farbe pre indikátor je potrebné pridať názov obrázka, ktorý je konvertovaný. A indikátor akcie, ktorá sa má vykonať. Preto argument bude rozšírený z reťazca na objekt. Objekt má názvy položiek a k nim pridané hodnoty. Napríklad `{color:"color", image:file.name, action:"action"}`. Prvá akcia bude `start`, ktorá označuje začiatok konverzie všetkých súborov a vynuluje element na zobrazenie stavu. Akcia `start` nemá priradený názov súboru a jeho farba je nepoužitá. Bude obsahovať zoznam `<ul>` s identifikátorom `status_list_field`. Ďalšia akcia je `file_start`, ktorá označuje začiatok konverzie samotného súboru. Musí mať názov súboru a aj farbu. Bude vytvorená nová položka zoznamu `<li>`, ktorá bude obsahovať názov obrázka a identifikátor stavu. Element zoznamu bude pridaný použitím jquery kombinácie `$(this).find("#status_list_field").append(item)`. `this` je odkaz na dialógové okno, a `find("#status_list_field")` nájde zoznam pre stavy obrázkov. Nakoniec `append(item)` pridá položku pre konkrétny obrázok. Identifikátor stavu je vytvorený ako kruh. Použitý na to bol element `<div>`. Jeho výška a šírka boli nastavené na veľkosť písma `width = height = 1em`. Preto jeho `border-radius` musí byť nastavený na `0.5em` aby vytvoril kruh. Na to, aby bol zarovno s textom jeho vlastnosť `display` musí byť nastavená na `inline-block`. Tento element bude jediná časť položky, ktorá sa bude meniť, preto jeho identifikátor je nastavený na názov obrázka. Indikátor je blízko k textu, preto mu bol nastavený `margin-left` na `0.5em`. Farba identifikátoru stavu bude nastavená pomocou jquery `$(this).find("div[id='"+arguments.image+"']").css("background-color", arguments.color)`. `this` je odkaz na dialógové okno. A `find("div[id='"+arguments.image+"']")` nájde identifikátor stavu pomocou jeho identifikátora. Nemôže byť použitá notácia selektoru s `#`, pretože názov obrázka obsahuje bodku. Bodka v tomto kontexte označuje triedu. Ak bude použití selektor `div#image.jpg`, tak bude hľadaný element `<div>` s identifikátorom `image` a s triedou `jpg`. Nakoniec `css("background-color", arguments.color)` nastaví farbu pozadia na zadanú farbu. Spracovanie sa rieši asynchrónne pomocou príkazu `setTimeout()` s krátkym časom. Spôsobí, že sa okno pri konverzii nezasekne.

8.4.2 Sprístupnenie obrázkov

V dialógu je ešte potrebné zobraziť obrázky, ktoré sú uložené v meta-dátach dokumentu. Na zobrazenie týchto obrázkov je možné použiť tabuľku. Ako hlavný údaj v tabuľke bude názov obrázka a zároveň aj názov položky v meta-dátach. Táto položka je použitá ako adresa pri odkazovaní na obrázok, ktorý je uložený v meta-dátach. K názvu položky je vhodné aj zo-

braziť ukážku obrázka. Ukážka obrázka je len orientačná. Preto stačí byť menších rozmerov. Napríklad kváder šírky `90px` a výšky `75px`. Ak by bola výška väčšia ako `75px`, tak by riadky tabuľky boli príliš veľké a do tabuľky by sa nezmestilo veľa riadkov bez rolovania. Ale nie je vhodné zmeniť pomer strán obrázka. Preto tieto parametre budú nastavené ako maximálna šírka `max-width` a výška `max-height`. Aj na bunku tabuľky budú nastavené rozmery. Šírka bude nastavená na `100px` a výška bude nastavená na `79px`. Tieto rozmery sú väčšie ako rozmery obrázka. Boli takto nastavené, aby sa obrázok zmestil do bunky, bez toho aby, ju zväčšil. V ďalšom stĺpci tabuľky bude base64 reťazec patriaci obrázku. Jeho účelom je možnosť skopírovať ho a použiť mimo rozšírenia. Base64 reťazec je dlhý, ale nie je potrebné ho zobraziť celý. Preto na zobrazenie môže byť použitý blok rovnakých rozmerov ako na ukážku obrázka. K správnemu zobrazeniu je ešte potrebné nastaviť pretečenie `overflow` na automatické `auto`. Ak sa text nezmestí do okna, tak bude base64 reťazec možné skrolovať. Base64 reťazec obsahuje dáta reprezentované ako text. Pretože base64 reťazec nie je prirodzený text, tak je vhodné nastaviť lámanie slov `word-break` na `break-all`. S týmto nastavením sa riadky nebudú lámať len na bielych znakoch a aj iných netextových znakoch. Ale budú vyzerať ako dáta a okno sa bude skrolovať iba jedným smerom. Dáta môžu byť označené pomocou kliknutia tri krát na text alebo použitím klávesovej skratky `ctrl+a`.

Tabuľka môže obsahovať viacero obrázkov. Viac než je možné zobraziť na obrazovke. Ak je dialógové okno väčšie ako stránka, tak samotnú stránku bude možné skrolovať. Lenže panel je staticky nastavený na vrchole stránky a skrolovať sa dá len telo s bunkami. Samotnú stránku normálne skrolovať nejde. Ak dialóg umožní stránku skrolovať, tak pod telom s bunkami bude prázdne miesto. A nástroje na pridanie obrázkov budú mimo zobrazenia. Preto je vhodné obmedziť veľkosť tabuľky s obrázkami. Obmedzenie veľkosti tabuľky bude vhodné obmedziť na násobok výšky riadu v tabuľke. Všetky riadky majú rovnakú výšku. Ako vhodná veľkosť by mohli byť štyri riadky. Preto bude tabuľka zabalená do elementu `<div>`. Jeho výška bude nastavená na `317px`. A jeho preplnenie na `auto`. Ak sa prekročí veľkosť tabuľky štyroch riadkov, tak nastavenie `auto` spôsobí, že tabuľku bude možné skrolovať.

Pridanie base64 reťazca a obrázka môže spôsobiť dlhé načítavanie dialógového okna. Bolo by vhodné spustenie dialógu optimalizovať. Náhľad obrázka a base64 reťazec je potrebné užívateľovi dodať, ale nie nevyhnutne pri každom spustení. Náhľad obrázka je potrebný častejšie ako base64 reťazec. Spustenie dialógového okna môže byť rýchlejšie ak budú obrázky načítané asynchrónne. Asynchrónne spustenie je najjednoduchšie pomocou funkcie `setTimeout()` s krátkym oneskorením. Base64 reťazec môže byť načítaný na vyžiadanie, namiesto implicitného načítania. Namieste base64 reťazca bude výzva na kliknutie. Keď bude bunka kliknutá, tak na miesto výzvy bude vložený base64 reťazec. Aj toto vloženie je vhodné spraviť asynchrónne pomocou `setTimeout()` s krátkym oneskorením. Kliknutie bude vyriešené nastavením reakcie na udalosť `onclick` nad bunkou tabuľky. V ďalšej bunke tabuľky budú tlačidlá pre funkcie s danými obrázkami. Tlačidlá v tejto bunke budú viazané na obrázok, pri ktorom sú. Obrázok, s ktorým budú funkcie pracovať, môže byť odkazovaný v atribúte. Napríklad atribút `data-src`. Na predanie názvu obrázka môže byť použitá aj funkcia `bind()`. Pomocou funkcie `bind()` môže byť predaný aj odkaz na dialógové okno, ktorý sa použije na vyvolanie udalostí nad ním. Odkaz na element sa nemôže predať v atribúte tlačidla.

Tabuľku je potrebné aktualizovať pri pridaní a odobratí obrázkov. Sledovať zmeny v dátach sa dá pomocou funkcie `watch()`. Zrušiť sledovanie sa dá pomocou funkcie `unwatch()`. Ale sledovať sa dá iba špecifický parameter objektu. Nedá sa sledovať vytvorenie objektu. Nedajú sa sledovať všetky parametre objektu a ani pridanie nového parametru. Preto sa fun-

kcia `watch()` nedá použiť na aktualizáciu tabuľky. Ďalším spôsobom, akým sa môžu sledovať zmeny v tabuľke je nastavenie intervalu. Interval sa nastavuje funkciou `setInterval()`. Tabuľka môže byť pri každom intervale prepísaná, ale takéto rozšírenie nie je elegantné a zbytočne zaberá zdroje. Tabuľka by sa mala prepísať len pri zmene údajov. Preto treba pri tiku intervalu zachytiť zmenu v objekte. Zachytenie zmeny v objekte sa dá urobiť vytvorením kópie objektu a porovnaním s reálnym objektom. Porovnanie objektov musí byť urobené manuálne. Tento systém je ale náročný na pamäť a aj na procesor, lebo base64 reťazce sú dlhé. Pretože práca s obrázkami sa vykonáva v rovnakom programe ako aj vyplnenie tabuľky, znovu vyplnenie tabuľky môže byť zavolané z funkcií, ktoré pracujú s obrázkami. Znovu vyplnenie tabuľky môže byť vytvorené ako udalosť. Udalosť musí byť viazaná k elementu. Element, na ktorý sa udalosť bude viazať, môže byť okno dialógu. Udalosť bude zaregistrovaná pri otvorení dialógu. Okno dialógu má špecifický identifikátor a pri otvorení dialógu sa na neho dá odkazovať pomocou `this`. Registrácia udalosti sa robí najjednoduchšie pomocou jquery. Registrácia sa dá urobiť pomocou jednej funkcie `bind()`. Prebehne použitím `$(this).bind('table_change', fill_image_table)`. Spustenie udalosti prebehne pomocou funkcie `$('#base_64_image_dialog').trigger('table_change', 'add')`. Žiadosť o znovu vyplnenie sa zavolá pri úspešnom pridaní obrázka cez textový vstup aj cez súborový vstup. A aj pri odstránení obrázka. Tabuľka sa nahradí použitím funkcie jquery `replaceWith()`. Tabuľka sa nájde pomocou jej identifikátora. Nahradenie tabuľky môže byť vyriešené kombináciou `$('#image_table').replaceWith(new table)`. Ale mohlo by to upraviť aj iné elementy stránky. Preto nemôžem ručiť za celú štruktúru stránky. Lepší spôsob vyhľadania tabuľky je použitie `$(this).find('#image_table').replaceWith(new _table)`. Funkcia `find()` nájde element určený jquery selektorom. Ale na rozdiel od predchádzajúceho riešenia nájdené elementy budú potomkovia elementu, nad ktorým bola vyvolaná udalosť znovu vyplnenia tabuľky.

8.4.3 Odstránenie obrázka

Užívateľovi je potrebné umožniť aj odstránenie obrázka. Ako na zvýšenie prehľadnosti v tabuľke, tak aj na zníženie veľkosti súboru. Na odstránenie obrázku z meta-dát je potrebný názov daného obrázku. Môže byť zadaný z textového vstupu. Napríklad pomocou funkcie `prompt()` alebo pomocou elementu `<input>` typu `text`. Narozdiel od vkladania obrázka pri odstránení obrázka je konečný počet platných vstupov, ktorý je daný počtom obrázkov uložených v meta-dátach. Preto na odstránenie obrázkov, na rozdiel od vkladania obrázkov, môže byť použitý rozbaľovací zoznam. Rozbaľovací zoznam musí byť naplnený názvami obrázkov z meta-dát. Ale ho bude nutné aktualizovať pri zmene dát, ktorá môže byť vyvolaná odstránením alebo vložením obrázka. Ako náhrada za rozbaľovací zoznam môže byť použitá tabuľka obrázkov. Tabuľka obrázkov obsahuje všetky obrázky z meta-dát a je aktualizovaná pri zmene. Je len potrebné zvoliť spôsob spustenia funkcie na odstránenie obrázka. A aj spôsobom predania názvu obrázka. Požiadavka na odstránenie obrázka sa dá zariadiť tlačidlom. Každý obrázok a tým pádom aj každý riadok tabuľky bude mať priradené tlačidlo na odstránenie obrázka, ktoré bude odpovedať danému obrázku. Názov obrázka na odstránenie môže byť funkcii, ktorá odstraňuje obrázky, predaný pomocou atribútu tlačidla alebo použitím funkcie `bind()`. Použije sa funkcia `bind()`. Umožňuje poslať funkcii nie len názov obrázka ale aj odkaz na dialógové okno.

Funkcia, ktorá odstraňuje obrázky, si najskôr overí platnosť názvu obrázka. Názov musí byť zadaný a musí existovať obrázok s daným názvom v meta-dátach dokumentu. Pri vybraní obrázku z tabuľky je nepravdepodobná neplatnosť názvu obrázka. Úložný priestor pre

obrázky v meta-dátach je realizovaný pomocou objektu. Obrázok sa odstráni z meta-dát použitím kľúčového slova `delete`. Kľúčové slovo `delete` odstráni vlastnosť objektu. Používa sa ako `delete expression`. `expression` je výraz, z ktorého by mal vzniknúť atribút objektu. Napríklad `delete object.property` alebo `delete object['property']` [8]. Ak bol obrázok odstránený, tak sa odkaz na dialógové okno použije na spustenie udalosti, ktorá aktualizuje tabuľku.

Kapitola 9

Záver

Vyrobené rozšírenia sú rozdelené do štyroch skupín. Ako rozšírenia spolupracujú medzi sebou je znázornené v obrázku 1.

Prvá skupina, s ktorou sa užívateľ môže stretnúť, sa zaoberá tvorbou dokumentu. Volá sa *Insert html and images* a jej identifikátor je `insert_html_img`. Obsahuje pomocné funkcie na prebranie materiálov z html dokumentov a vloženie obrázkov. Html dokument na vloženie môže byť webová stránka alebo lokálny html dokument. Môže byť vložený celý, alebo len jeho časť. Ak sa vkladá menšia časť html dokumentu, tak je možné použiť schránku clipboard. Pretože systém Jupyter pracuje ako webové rozhranie, obrázky musia byť uložené mimo samotného dokumentu, ale pomocné funkcie na vloženie obrázkov to môžu zmeniť. Umožňujú uložiť obrázok v meta-dátach dokumentu a následné zobrazenie v dokumente.

Druhá skupina rozšírení, s ktorou sa môže užívateľ stretnúť, sa zaoberá prípravou dokumentu. Volá sa *Common utilities* a jej identifikátor je `jupyter_common_utils`. Obsahuje sa základné grafické úpravy, ako úprava písma pre kód a zobrazenie kombinácie odkazu a kódu vo formátovanom texte. Ďalej obsahuje prípravu dokumentu na použitie, ako hromadné priradenie typu snímky a vyčistenie výstupu niektorých buniek.

Najčastejším použitím dokumentu bude jeho prezentácia. Ak sa dokumentu bude prezentovať použitím rozšírenia RISE, tak sa užívateľ môže stretnúť s treťou skupinou rozšírení. Volá sa *RISE configuration* a jej identifikátor je `rise_config`. Zaoberá sa nastavením parametrov pre rozšírenie RISE, ako pre samotné rozšírenie, tak aj pre jednotlivé snímky, a aj pridaním ďalšej funkcionality pre zjednodušenie samotnej prezentácie.

Štvrtá skupina rozšírení sa zaoberá konvertovaním dokumentu do LaTeX formátu. Skladá sa z rozšírenia *Compare code*, ktorý umožňuje pridať značky pre konvertovanie do dokumentu. Obsahuje skript, ktorý zásadne upraví proces konverzie do formátu LaTeX. A predprocesor patriaci rozšíreniu *Common utilities*, ktorý niektoré grafické zmeny prevedie aj do formátu LaTeX.

Automatizované vkladanie materiálov zásadne zrýchli vytváranie dokumentu. Bez neho je potrebné vytvoriť každú bunku zvlášť a potom do nej skopírovať správny text. S použitím automatizovaného vkladania je tento proces skrátený na zopár úkonov. Text bude do buniek rozdelený automaticky. Hromadné pridanie typu snímky takisto zásadne zrýchli vytváranie dokumentu pri väčších dokumentoch. Namiesto manuálneho priraďovania typu snímky každej bunke, typ snímky je priradený iba kľúčovým bunkám a zvyšným bunkám bude typ snímky priradení s dvoma kliknutiami. Úprava písma pre kód ho urobí jednoduchší na čítanie. Zobrazenie kombinácie odkazu a kódu vo formátovanom texte nebolo jednoduché čítať, preto jeho úprava zjednoduší jeho čítanie. Automatizovaná príprava dokumentu pripraví dokument na jeho prezentáciu pomocou jedného kliknutia podľa užívateľovho nastavenia.

Nastavenie prostredia na prezentáciu umožní začať prezentáciu na vybranej bunke a po ukončení prezentácie sa vrátiť na miesto odkiaľ skončila. Uloženie obrázkov v dokumente zjednoduší jeho prenesenie, pretože stačí preniesť iba jeden súbor. A úprava konvertovania sprehľadní výsledný pdf dokument, tak ako aj jeho zdrojový súbor.

Literatúra

[1] Albinson, P.: *Enhancing Programming Lectures Using Interactive Web-Based Lecture Slides.* Annual Workshop of the Psychology of Programming Interest Group, *ročník 27, 9 2016: s. 7–10, st. Catharine's College, University of Cambridge, UK.*

[2] damianavila: RISE. [Online; navštíveno 7.11.2016].
URL https://github.com/damianavila/RISE

[3] Frederic, J.: Writing IPython Notebook Plugins. [Online; navštíveno 7.11.2016].
URL https://www.safaribooksonline.com/blog/2013/12/18/ipython-notebook-plugins/

[4] hakimel: reveal.js. [Online; navštíveno 7.11.2016].
URL https://github.com/hakimel/reveal.js/blob/dev/README.md#slide-transitions

[5] Jupyter Development Team: Customizing nbconvert. [Online; navštíveno 24.10.2016].
URL http://nbconvert.readthedocs.io/en/latest/customizing.html

[6] Jupyter Development Team: nbconvert Documentation. [Online; navštíveno 28.10.2016].
URL https://media.readthedocs.org/pdf/nbconvert/latest/nbconvert.pdf

[7] Jupyter Development Team: Preprocessors. [Online; navštíveno 24.10.2016].
URL http://nbconvert.readthedocs.io/en/latest/api/preprocessors.html

[8] Mozilla Developer Network: delete operator - JavaScript. [Online; navštíveno 28.4.2017].
URL https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/delete

[9] Roels, R.; Signer, B.: MindXpres: An Extensible Content-Driven Cross-Media Presentation Platform. In Web Information Systems Engineering — WISE 2014, Lecture Notes in Computer Science, *ročník 8787, editace B. B.; B. A.; M. Y.; V. A.; Z. Y., Springer, Cham, 2014.*

[10] Roganov, E. A.; Roganova, N. A.; Aleksandrov, A. I.; aj.: Web portal for dynamic creation and publication of teaching materials in multiple formats from a single source representation. In AIP Conference Proceedings, *ročník 1797, American Institute of Physics, 2017, doi:10.1063/1.4972436.*

[11] W3C Web Security: Same Origin Policy. [Online; navštíveno 2.12.2016].
URL https://www.w3.org/Security/wiki/Same_Origin_Policy

[12] *W3Schools: HTML Encoding (Character Sets). [Online; navštíveno 22.12.2016].*
URL http://www.w3schools.com/html/html_charset.asp

[13] *W3Schools: HTML <input> multiple Attribute. [Online; navštíveno 27.4.2017].*
URL https://www.w3schools.com/tags/att_input_multiple.asp

[14] *WWW stránky: How can I highlight some lines from source code? [Online; navštíveno 7.11.2016].*
URL http://tex.stackexchange.com/questions/8851/how-can-i-highlight-some-lines-from-source-code

[15] *WWW stránky: How do I set custom CSS for my IPython/IHaskell/Jupyter Notebook? [Online; navštíveno 27.9.2016].*
URL http://stackoverflow.com/questions/32156248/how-do-i-set-custom-css-for-my-ipython-ihaskell-jupyter-notebook

[16] *WWW stránky: How to draw circle in html page? [Online; navštíveno 27.4.2017].*
URL http://stackoverflow.com/questions/6921792/how-to-draw-circle-in-html-page

[17] *WWW stránky: Jupyter Nbextensions Configurator. [Online; navštíveno 11.10.2016].*
URL https://github.com/Jupyter-contrib/jupyter_nbextensions_configurator

[18] *WWW stránky: Jupyter notebook extensions. [Online; navštíveno 11.10.2016].*
URL https://github.com/ipython-contrib/jupyter_contrib_nbextensions

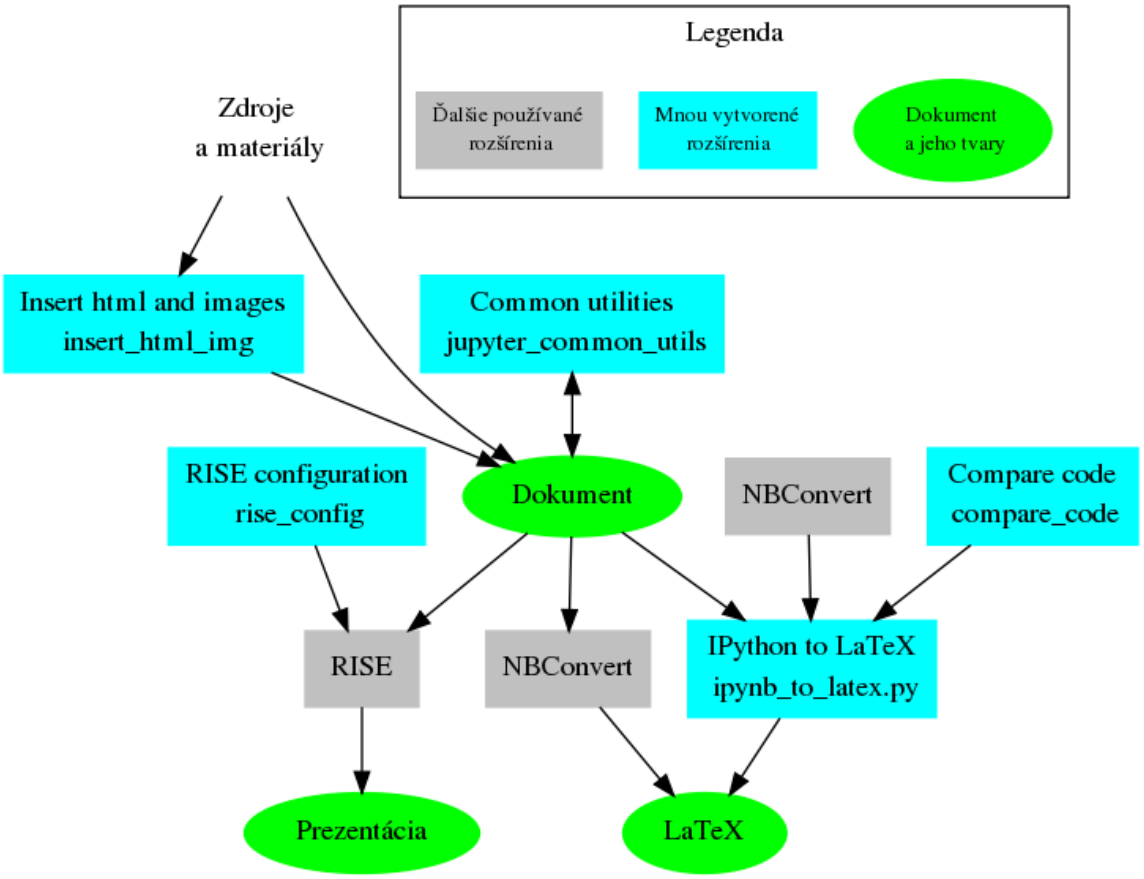
[19] *WWW stránky: LaTeX/Source Code Listings. [Online; navštíveno 7.11.2016].*
URL https://en.wikibooks.org/wiki/LaTeX/Source_Code_Listings

[20] *WWW stránky: Python win32clipboard Examples. [Online; navštíveno 1.5.2017].*
URL http://www.programcreek.com/python/index/3337/win32clipboard

[21] *WWW stránky: Skipping line numbers in lstlisting. [Online; navštíveno 7.11.2016].*
URL http://tex.stackexchange.com/questions/264361/skipping-line-numbers-in-lstlisting

Prílohy

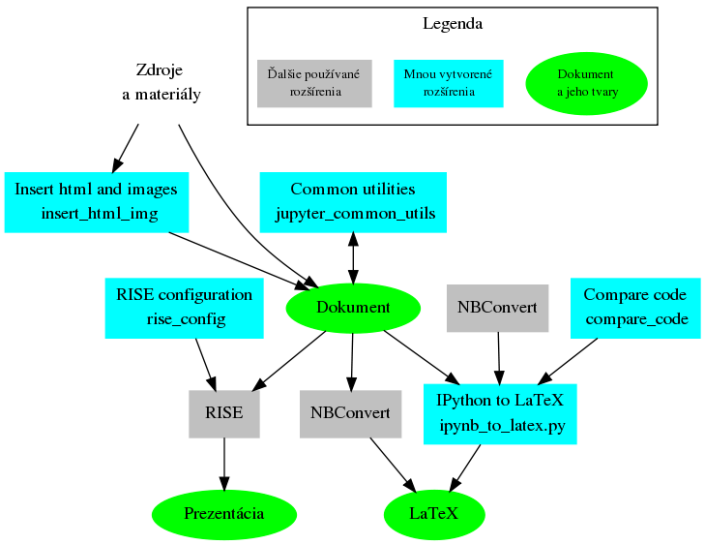
Obrázky



Obr. 1: Interakcia rozšírení

Plagát

Interakcia rozšírení



Insert html and images `insert_html_img`

Vstup:
- url k stránke
- cesta k súboru
- prázdny alebo "clipboard" pre vstup zo schránky clipboard

Definuje konvertor `insert_html_img/convert.py`
Získa izolačný reťazec xpath pre známe stránky z konfigurácie rozšírenia
Vytvorí bunku na spustenie konvertora

Spustí dialógové okno na spravovanie obrázkov
Obrázky uložené v meta-dátach dokumentu v base64

Common utilities `jupyter_common_utils`

Grafická úprava kombinácie odkazu a kódu
Nastavenie fontu pre kód: Roboto Mono

Spustí markdown bunky
Spustí `%%html` bunky
Spustí bunky, ktoré sa začínajú z "`# Run`"
Vyčistí výstupy ostatných kódových buniek

Vyčistí všetky výstupy buniek

Dialógové okno s výberom typu snímky.
Typ snímky sa nastaví pre bunky, ktoré ho nemajú nastavené

Označí všetky bunky

RISE configuration `rise_config`

Prepína Cell Toolbar medzi: Slideshow → Slide transition → Slide transition speed → Hide in rise → Slideshow

Vloží bunku typu javascript "`%%javascript`" na nastavenie RISE
Údaje v bunke sú z nastavenia v dokumente

Vloží bunku typu javascript "`%%javascript`" na nastavenie RISE
Údaje v bunke sú z nastavenia rozšírenia, ktoré sa dá zmeniť v Jupyter Nbextensions Configurator.

Po návrate z RISE prejde na vybranú bunku namiesto začiatku dokumentu
Cell Toolbar → Slide transition: zmení prechod pre konkrétnu bunku
Cell Toolbar → Slide transition speed: zmení rýchlosť prechodu pre konkrétnu bunku
Cell Toolbar → Hide in rise: schová určitú časť pri prezentácií

Compare code `compare_code`

View → Cell Toolbar → Compare code
Nastavenia pre konverziu

IPython to LaTeX `ipynb_to_latex.py`

Pracuje v Python3
python `ipynb_to_latex.py file.ipynb`