Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství


Diplomová práce

# Nástroj pro validaci zápisu kaskádových stylů ve webovém prohlížeči

Bc. Martin Krištof

# ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

# ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Martin Krištof

Informatika

Název práce

**Nástroj pro validaci zápisu kaskádových stylů ve webovém prohlížeči**

Název anglicky

**Cascading style sheets validation tool in web browser**

---

**Cíle práce**

Práce je zaměřena na problematiku zpracování CSS. Cílem práce je navrhnout a implementovat nástroj jakožto rozšíření do webového prohlížece pro validaci zápisu kaskádových stylů využívající pravidla definovaná podle metodiky BEM (Block Element Modifier). Dílčím cílem bude poskytnout přehled použitých technologií.

**Metodika**

Teoretická část diplomové práce je založena na studiu odborných informačních zdrojů. Na základě zjištěných poznatků budou popsány teoretická východiska nutná pro zpracování aplikace.

V navazující praktické části práce bude provedena analýza stávajících nástrojů pro validaci CSS na jejímž základě bude navržen a implementován nástroj sloužící k validaci zápisu kaskádových stylů na základě pravidel definovaných podle metodiky BEM (Block Element Modifier). Výstupem bude aplikace, která bude fungovat v prohlížeči jako jeho rozšíření. Nástroj bude otestován, postup jeho vývoje popsán a zhodnocen a budou navrženy případné možnosti jeho dalšího rozvoje.

Při zpracování praktické části budou využity standardní metody a nástroje softwarového inženýrství.

**Doporučený rozsah práce**

60-80 stran

**Klíčová slova**

CSS BEM Linter Javascript Node.js Chrome Extension

---

**Doporučené zdroje informací**

BHATIA, Ashish Singh. Using React JS for Web Applications. Open Source for You [online]. 2016, 4(8), 56-58 [cit. 2018-03-17]. ISSN 09741054.

HOLZNER, S. *JavaScript : profesionálně : [kompletní referenční příručka].* Praha: Mobil Media, 2003. ISBN 80-86593-40-1.

MANTYLA, Dan. Functional Programming in JavaScript. 2015. Packt Publishing, 2015. ISBN 9781784398224.

MEYER, E A. – MEYER, E A. *CSS : the definitive guide.* Beijing ; Sebastopol, CA: O'Reilly, 2007. ISBN 0596527330.

PERGL, R. – VANÍČEK, J. *Mathematical theory of programs : part 3: logical and functional programming paradigms..*

---

**Předběžný termín obhajoby**

2018/19 LS – PEF

**Vedoucí práce**

Ing. Jiří Brožek, Ph.D.

**Garantující pracoviště**

Katedra informačního inženýrství

Elektronicky schváleno dne 24. 1. 2019

**Ing. Martin Pelikán, Ph.D.**

Vedoucí katedry

Elektronicky schváleno dne 24. 1. 2019

**Ing. Martin Pelikán, Ph.D.**

Děkan

V Praze dne 03. 02. 2019

**Čestné prohlášení**

Prohlašuji, že svou diplomovou práci "Nástroj pro validaci zápisu kaskádových stylů ve webovém prohlížeči" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 22.3.2019

**Poděkování**

Poděkování patří panu Ing. Jiřímu Brožkovi, Ph.D. za vedení práce a Robinu Pokornému za ideu a koordinaci metod.

# Nástroj pro validaci zápisu kaskádových stylů ve webovém prohlížeči

**Abstrakt**

Závěrečná práce navazuje na moji bakalářskou práci a zabývá se vytvořením dostupnějšího validačního nástroje v podobě rozšíření do prohlížeče pro zápis kaskádových stylů (dále CSS) dle metodiky zvané Block Element Modifier (dále BEM).

Obsahuje teoretická východiska a úvod pro použitých technologií při tvorbě tohoto nástroje.

Pro vývoj byl zvolen programovací jazyk JavaScript v podobě ECMAScript 2018, architektura web extensions a knihovny React a Redux. Tyto technologie jsou v práci popsány.

Výsledkem této práce je rozšíření do webových prohlížečů, které lze po instalaci spustit na libovolné webové stránce.

Samotná aplikace je pojmenována jako *BEM-validator-extension* a publikována jako *BEM Validator*.

**Klíčová slova:** CSS BEM Linter Javascript Node.js Chrome Extension

# Cascading style sheets validation tool in web browser

**Abstract**

This thesis builds on my bachelor thesis and deals with creation of a more accessible validation tool in the form of a browser extension for cascading styles naming (CSS) according to the methodology called Block Element Modifier (BEM).

It contains theoretical starting points and an introduction to the technologies used during the development of this tool.

ECMAScript 2018 version of JavaScript programming language was chosen for development. Another used technologies like web extensions architecture, and libraries React and Redux are described in the first part the of thesis.

The result of this all is a web extension for browsers which could be run on any visited web page after installing. The application itself is named *BEM-validator-extension* and published as *BEM Validator*.

**Keywords**: CSS BEM Linter Javascript Node.js Chrome Extension

# Obsah

# Seznam obrázků

# Seznam tabulek



# Seznam použitých zkratek

AMO, 49
API, 13
AST, 7
BEM, 6
CLI, 87
CSP, 27
CSS, 6
DOM, 21
HTML, 20
IDE, 90
JSON, 39
JSX, 51
NPM, 8
SVG, 67
TDD, 86
UI, 31
URI, 21
URL, 29
VDOM, 52
XHR, 21

# 1 Úvod

Výsledným produktem závěrečné práce s názvem ***„Nástroj pro validaci zápisu kaskádových stylů ve webovém prohlížeči“*** je aplikace pojmenována jako *BEM-validator-extension* pod MIT licencí a je publikována jako *BEM Validator* v repozitáři Mozilla Firefox doplňků.

Tato aplikace slouží pro kontrolu existujících tříd kaskádových stylů na libovolné webové stránce, a to podle metody BEM.

Aplikace využívá existující modul *CSS-should-plugin-bem*[1], který jsem vytvořil jako závěrečnou práci bakalářského studia na ČZU sloužící jako nástroj typu linter (IT Slovník, 2017). Existující řešení bylo nutné vylepšit a upravit tak, aby bylo možno komunikovat s aplikací. Zasahoval jsem tedy do obou projektů.

BEM je konvence pojmenovávání tříd v CSS, které se důkladně věnuji ve své bakalářské práci.

Lintery podobného typu, který jsem v minulosti vytvořil, už existují. Žádný z nich ale není součástí rozšíření do webového prohlížeče.

V závěru práce zhodnocuji možná vylepšení aplikace.

Aplikaci lze používat nejen pro studentské účely, ale zejména pro účely profesní, protože její vznik vycházel právě z praxe. V komunitě vývojářů BEM hojně používáme a tento nástroj by všem usnadnil kontrolu jejich vlastní práce.

### Volba tématu

Už od začátku psaní bakalářské práce jsem počítal s tím, že na ni budu v magisterském studiu navazovat. Chtěl jsem vytvořit rozšíření do prohlížeče a moje téma závěrečné práce se pro uskutečnění nabízí. Nápad, že vznikne tato aplikace tedy přesahuje několik let zpět.

[1] Odkaz na repozitář s projektem: https://github.com/MartinKristof/css-should-plugin-bem

Aplikace pomůže i mým kolegům v profesním životě. Nebudou muset složitě integrovat na serveru původní nebo konkurenční nástroj – jen si jednoduše stáhnou a nainstalují vytvořenou aplikaci do prohlížeče a výsledky analýzy vidí okamžitě.

Tím, že je projekt umístěn ve veřejném repositáři na Githubu, se může každý zapojit do přidávání vylepšení či oprav chyb.

# 2 Cíl práce a metodika

Práce je rozdělena na dvou částí – teoretické a praktické.

V první části je teoretický popis metodiky BEM, přehled existujících desktopových webových prohlížečů, charakteristika web (browser) extensions a knihovny React, a základní informace o abstraktním syntaktickém stromu (AST) a architektuře pro udržování stavu JavaScriptových aplikacích (Redux).

Druhá část popisuje návrh řešení včetně grafického návrhu aplikace. Dále se věnuje implementaci samotné aplikace pomocí nástrojů uvedených v teoretické části této práce. Dokumentuje teoretická východiska souvisící s teoretickou částí práce.

## 2.1 Cíl práce

### 2.1.1 Hlavní cíl

Hlavní cíl práce je analýza, návrh a implementace webového rozšíření do prohlížeče jakožto validačního nástroje pro zápis tříd kaskádových stylů dle metodiky *Block Element Modifier*. Mimo to je nedílnou součástí poskytnout vhled do použitých technologií.

Aplikace bude schopna na navštěvované stránce zkontrolovat existující CSS třídy v dokumentu oproti metodice BEM a vypíše uživateli informaci o této kontrole.

V případě chyby zobrazí seznam selektorů, u kterých je chyba s možností prokliku na jejich detail v nativním panelu prohlížeče. V případě validity nebo nenalezení tříd, které jsou dle metodiky psány, rovněž zobrazí informaci o tomto stavu.

### 2.1.2 Dílčí cíle

Dílčími cíli závěrečné práce jsou:

- popis metodiky BEM,
- charakteristika fungování web extension,
- testování doplňků,
- představení knihoven React a Redux.

## 2.2 Metodika

Řešení vychází z metodiky Unified Process. Především jde o iterativní vývoj, tak aby bylo možno aplikaci průběžně testovat. K vývoji bylo využito i metodiky TDD.

Postup lze rozdělit na jednotlivé etapy:

- stanovení akceptačních kritérií,
- analýza a navržení funkcionality,
- návrh grafického rozhraní,
- studium web extensions,
- popis integrace s existujícím validačním nástrojem,
- implementace a testovaní,
- vytváření instalačního balíčku,
- publikování do NPM[2],
- publikování do repozitáře pro doplňky.

Publikace bude zatím provedena pouze pro prohlížeč Mozilla Firefox z důvodů, které jsou zmíněny v další kapitole. Do Chrome webstoru se doplněk dostane později, už mimo tuto závěrečnou práci. Tento krok není nezbytný pro splnění zadání práce.

[2] O NPM na: https://docs.npmjs.com/getting-started/what-is-npm

# 3 Teoretická východiska

## 3.1 Block Element Modifier – BEM

*Block Element Modifier* – BEM je pojmenovávací konvence pro třídy v kaskádových stylech. Je to způsob, jak pojmenovat třídy v CSS, aby nedošlo k míchání různých typů tříd (Michálek, 2017).

BEM byl vytvořen v ruské firmě Yandex[3] v roce 2005. Hlavní myšlenka spočívá v rozdělení tříd do tří skupin:

| ***Typ třídy*** | ***Způsob pojmenování*** |
|---|---|
| ***blok*** | **.block** |
| ***element*** | **.block__element** |
| ***modifikátor*** | **.block__modifier** |

**Tabulka 1: BEM skupiny (zdroj: vlastní)**

Potomek bloku je element, modifikátor je pak variantou konkrétního elementu. Jde tedy o logické rozdělení zanořených elementů (selektorů) uvnitř.

Tato metodika pojmenování se hojně používá ve světě vývoje, jelikož je tak kód na front-endu[4] přehlednější a srozumitelnější. CSS třídy jsou totiž rovněž viditelné uvnitř HTML kódu a tím se právě ona čitelnost projevuje. Tento způsob velmi usnadňuje týmovou práci na projektu. (Krištof, 2017)

Kromě této domény přidává tato metodika ještě benefit rozšiřování zápisu již existujících tříd. K tomu se hodí modifikátory, které nějakým způsobem upravují již definovanou komponentu. Například může modifikátor měnit barvu, písmo, pozici a další libovolné CSS vlastnosti. (Krištof, 2017)

[3] Webová stránka firmy https://www.yandex.ru/
[4] Definice slova front-end: http://it-slovnik.cz/pojem/frontend

```
<!DOCTYPE html>
<html lang="cs">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <div>
        <ul class="nav " role="menu">
            <li class="nav__item nav__item--muted">
                <a href="/">Úvod</a>
            </li>
            <!-- ... -->
        </ul>
    </div>
</body>
</html>
```

- **.nav** je blok (název komponenty)
- **.nav__item** je element (potomek komponenty)
- **.nav__item—muted** je modifikátor (varianta elementu)

```
.nav {
    display: flex;
}
.nav__item {
    padding: 10 20px;
}
.nav__item--muted {
    background-color: #ccc;
}
```

První třída ***nav*** definuje styly pro celý blok (komponentu). Druhá třída v pořadí ***nav__item*** je pak elementem komponenty a představuje jednotlivý prvek v celku (v tomto případě seznamu). Třída ***nav__item—muted*** je modifikátorem výše zmíněného elementu. Modifikuje zobrazení a to tak, že nastavuje barvu pozadí.

Tím, že se používá obecné pojmenování, je pak hned z kódu jasné, k čemu je daná třída určena. Nesmí se ale pod jednu třídu logicky kombinovat více vlastností, aby pak vlastnosti třídy nedefinovaly více funkcionality, než je sémanticky záhodno. (Krištof, 2017)

Například třída ***nav—item__muted*** nesmí obsahovat CSS pro pozicování, velikost písma nebo viditelnost. Tyto vlastnosti by měla definovat jiná třída, která by mohla být znovupoužitelná a aplikována na jiných elementech. (Krištof, 2017)

Pokud chceme nastavit libovolnému elementu červenou barvu textu, vytvoříme další třídu a té tuto vlastnost nastavíme. Třídu ale pojmenujeme obecně jako ***text—warning,*** aby se vystihlo chování třídy, která upozorňuje v kontextu na něco varovného. Toto je další využití modifikátoru a předpokládáme, že už existuje třída ***text***, která definuje vlastnosti

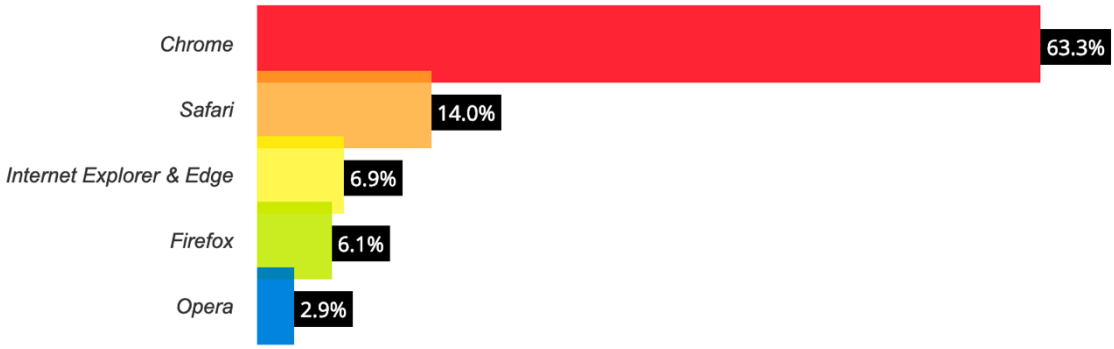
textu (velikost, šířku písma, styl písma apod.). Pokud by třída ***text*** neexistovala, jde o špatné použití modifikátoru, jak je zmíněno výše, i o porušení konvence. Modifikátor může být definován pouze pro již existující element.

BEM metodiku lze celkem jednoduše aplikovat a lehce se ji naučit, tudíž se z ní stává standard.

Preprocesory[5] umí usnadnit BEM zápis. Jde o zanořování dalších tříd do sebe s využitím proměnné v bloku. (Krištof, 2017)

## 3.2 Nejrozšířenější desktopové webové prohlížeče

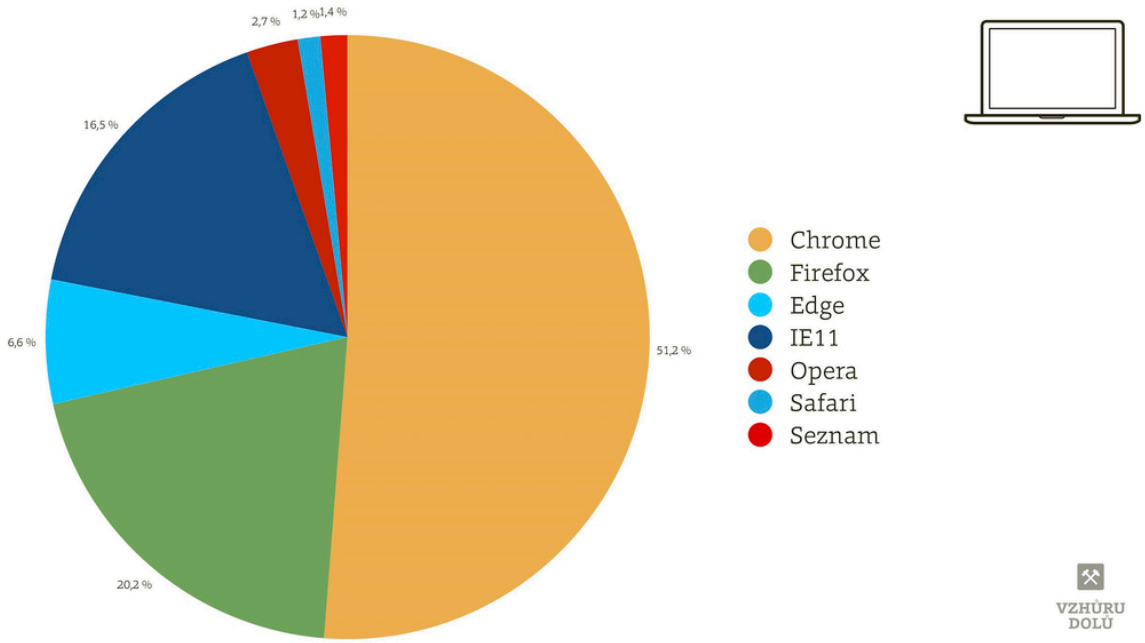
Nejrozšířenější a nejpoužívanější webové prohlížeče[6] patří celosvětově několik produktů. Chrome, Firefox, Opera, Safari, Internet Explorer, Microsoft Edge, Chromium a mnohé další. Údajně jich existuje minimálně 22 a to navíc v různých verzích[7].




**Obrázek 1: Statistiky používání desktopových prohlížečů leden 2019 (zdroj: Google Chrome, 2018)**

[5] Preprocesory: http://www.vzhurudolu.cz/blog/12-css-preprocesory-1
[6] Webové prohlížeče https://www.jaknainternet.cz/page/1235/prohlizece-a-internetove-technologie/
[7] Přehled prohlížečů https://www.zive.cz/clanky/nasli-jsme-22-schopnych-internetovych-prohlizecu-vyberte-si-ktery-vam-nejvic-sedne/sc-3-a-193186/default.aspx#part=1

V České republice pak statistika vypadá následovně:




**Obrázek 2: Statistická data používání prohlížečů v ČR z roku 2018 (zdroj: https://www.vzhurudolu.cz/prirucka/prohlizece )**

Z grafů vyplývá, že jednoznačně nejpoužívanější prohlížeče jsou Chrome, Safari, Firefox, IE/Edge. S menším podílem s používají prohlížeče Opera, Safari nebo Seznam.

Dalšími existujícími prohlížeči jsou například produkty Vivaldi či Brave.

## 3.3 Web extensions (Browser extensions)

### *3.3.1 Základní charakteristika*

Aby bylo možno vytvořit aplikaci, která poběží uvnitř webového prohlížeče jako jeho rozšíření, je potřeba se zaměřit na fungování a komunikaci jednotlivých vrstev aplikace.

Jde o malé aplikace, které nějakým způsobem rozšiřují zážitek uživatele. Tyto aplikace mohou přidávat funkcionalitu, modifikovat obsah v prohlížeči. K tomu využívají technologie jako HTML, CSS a JavaScript. (Google Chrome, 2018)

Web extensions jsou kompatibilní webová rozšíření napříč novějšími webovými prohlížeči jako Firefox, Opera, Chrome, Vivaldi, Brave, Chromium i Microsoft Edge. Safari toto rozhraní zatím nepodporuje. (Mozilla, 2018)

V roce 2017 vznikl první komunitní draft pro W3C Community and Business Groups[9]. Dříve se totiž pro výše zmíněné prohlížeče musela vytvářet specifická webová rozšíření, která nebyla navzájem kompatibilní.

Bylo tedy potřeba udržovat několik verzí stejného doplňku pro více prohlížečů. Tento stav nebyl pro vývojáře příznivý, jelikož vývoj, testování a údržba více verzí těchto aplikací je velmi nákladná a neefektivní. (Browser Extension Community Group , 2017)

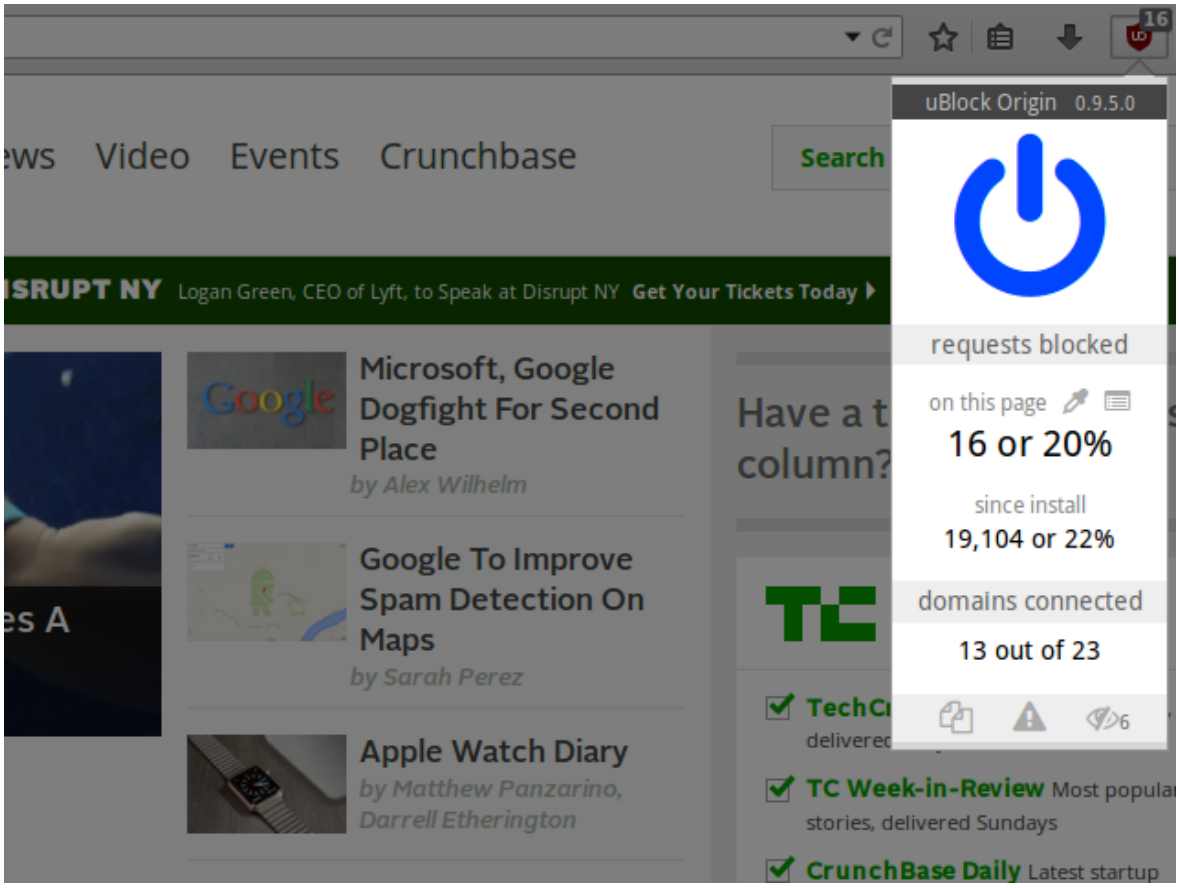
Díky *web extensions* API lze aplikaci (webové rozšíření) napsat jen jednou a s drobnými změnami pak používat v prohlížečích, které toto rozhraní podporují. Jde tedy o pokus o sjednocení.

Zatím se tedy nejedná o standard, ale lze jej používat v novějších verzích prohlížečů. (Mozilla, 2018)

### 3.3.2 Využití web extensions

Existuje velmi mnoho případů použití těchto aplikací. Velmi známá jsou rozšíření pro blokování obsahu, konkrétně rušivých reklam a vyskakovacích oken Adblock[10] nebo uBlock[11]. Tento typ patří do skupiny rozšíření, která přidávají, či odebírají obsah z webových stránek. Dalším možností je přeformátování HTML či CSS na stránce, tak aby měl uživatel lepší zážitek z konzumování obsahu, který jej skutečně zajímá. (Mozilla, 2018)

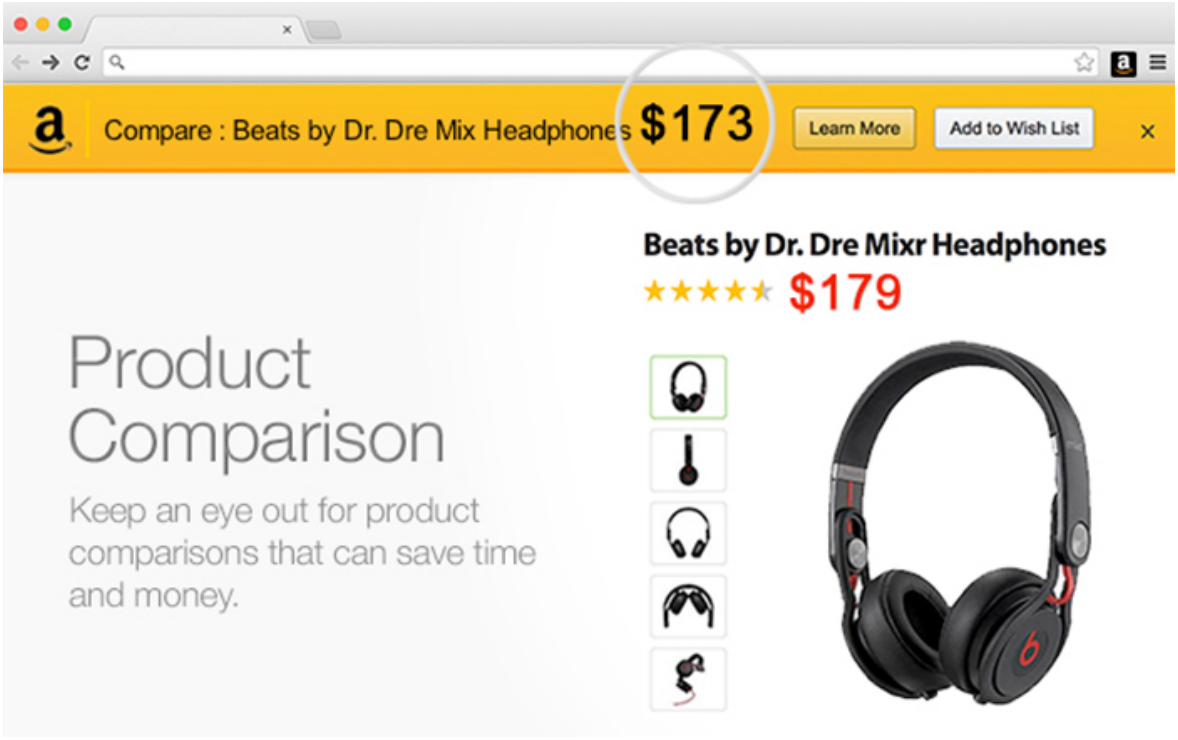
[9] O komunitě https://www.w3.org/community/
[10] Odkaz na aplikaci https://adblockplus.org/
[11] Odkaz na doplněk https://addons.mozilla.org/en-US/firefox/addon/ublock-origin/




**Obrázek 3: Ukázka doplňku uBlock**
**(zdroj: https://addons.mozilla.org/en-US/firefox/addon/ublock-origin/)**

Dalším druhem je rozšiřovací či doplňovací funkcionalita webové stránky. Pomocí doplňku lze dodat další funkce nebo informace do prohlížeče z webu tvůrce. Tak lze uživatelům shromažďovat detaily ze stránek, které navštěvují, a vylepšit službu, kterou provozovatel nabízí. Například Amazon Assistant for Firefox[12]. (Mozilla, 2018)

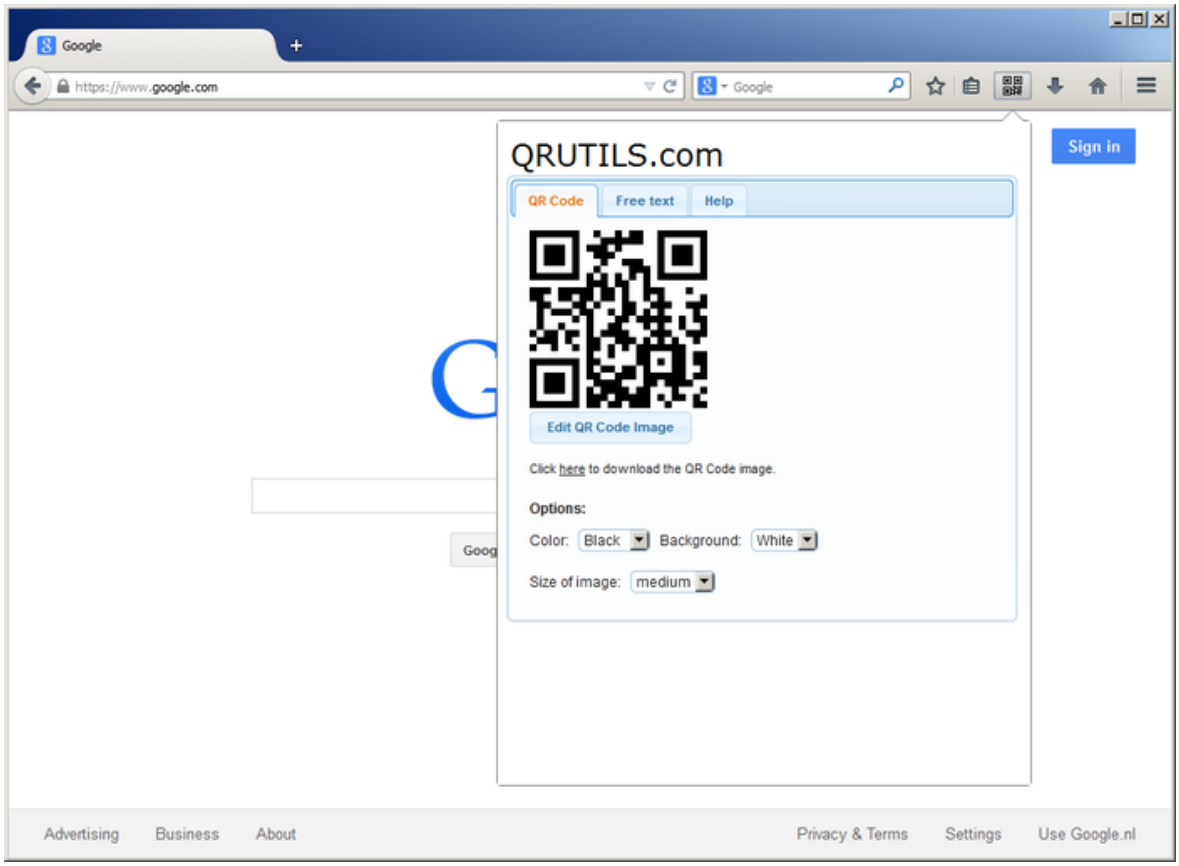
[12]Odkaz na doplněk https://addons.mozilla.org/en-US/firefox/addon/amazon-browser-bar/




**Obrázek 4: Ukázka doplňku Amazon browser bar**
**(zdroj: https://addons.mozilla.org/en-US/firefox/addon/amazon-browser-bar/)**

Další skupinou jsou nástroje a nové funkce při prohlížení webu. Jedná se například o generátory QR kódů z URL adres, odkazů nebo textů stránek (QR Code Image Generator[13]). Lze tedy přidat do prohlížeče nové funkce a využívat je na kterékoliv webové stránce. (Mozilla, 2018)

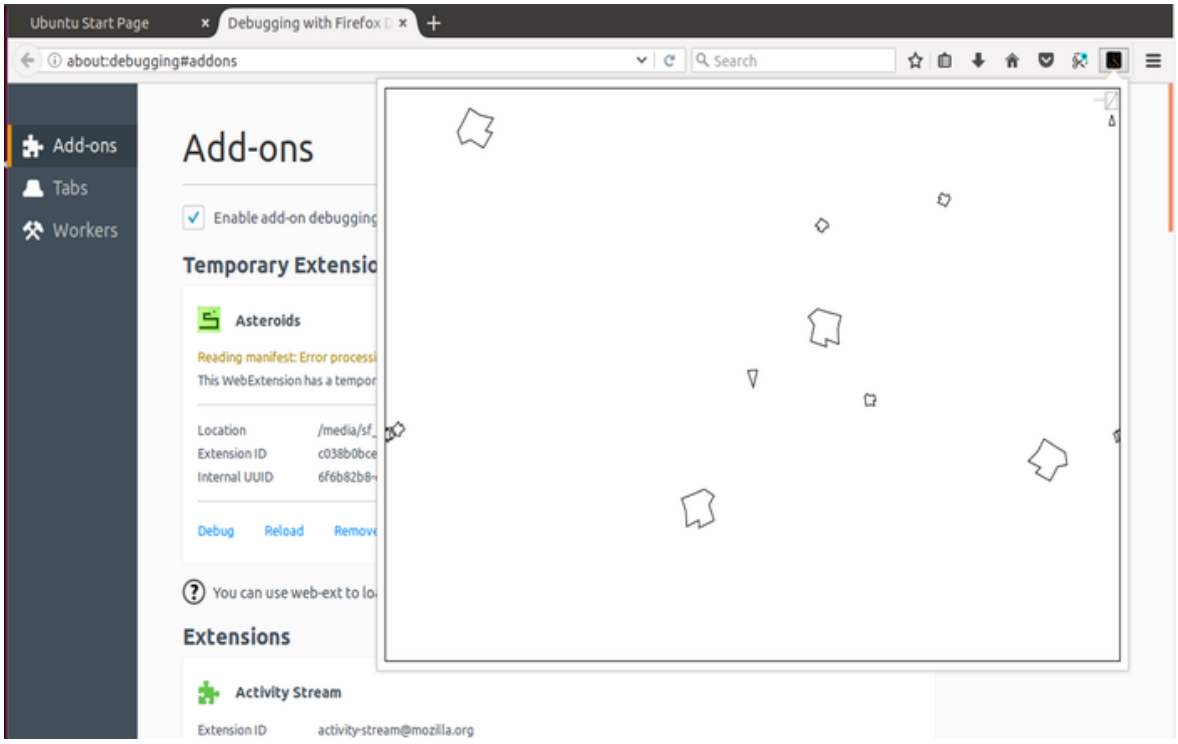
[13]Odkaz na doplněk https://addons.mozilla.org/en-US/firefox/addon/qr-code-image-generator/




**Obrázek 5: Ukázka doplňku QR code image generator (zdroj: https://addons.mozilla.org/en-US/firefox/addon/qr-code-image-generator/)**

Zábavnější skupinou jsou hry. Lze takto nabídnout uživateli možnost si zkrátit čekání ve virtuální frontě na vstupenku nebo prostě jen tak napsat hru v JavaScriptu a spouštět ji právě v prohlížeči a to off-line. Příkladem je hra Asteroids in Popup[14]. (Mozilla, 2018)

[14] Odkaz na doplněk https://addons.mozilla.org/en-US/firefox/addon/asteroids-in-popup/



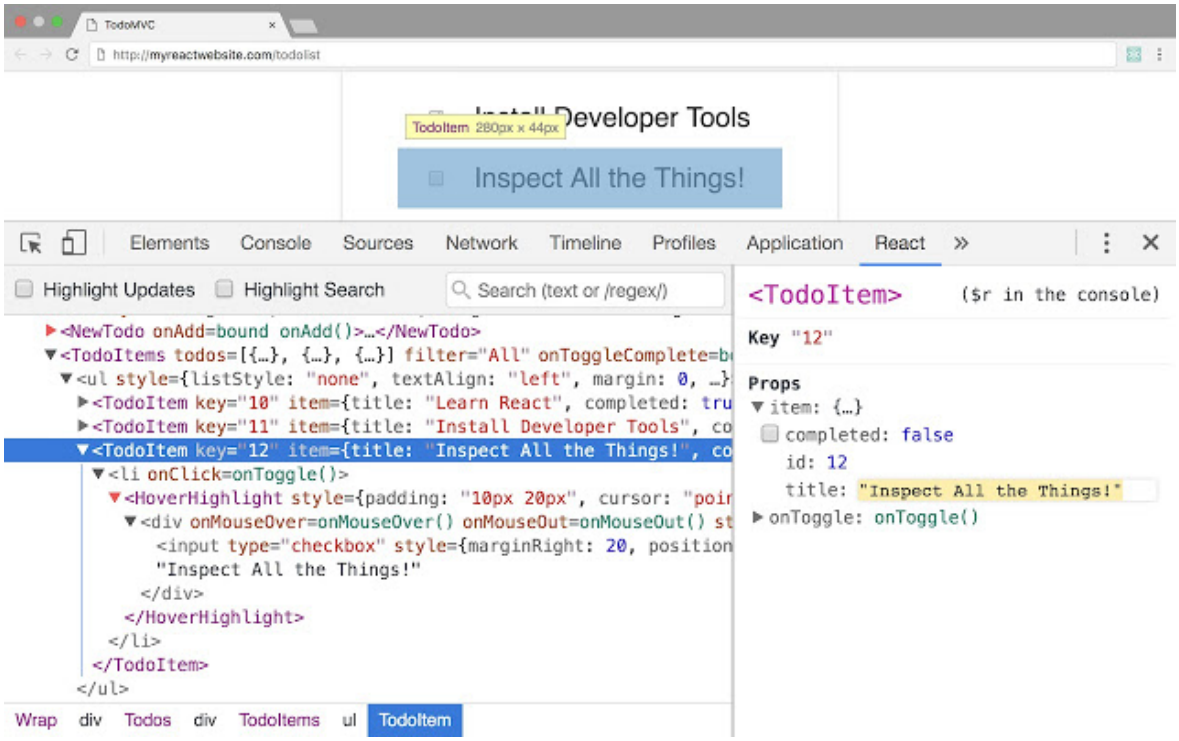

**Obrázek 6: Ukázka doplňku hry Asteroid in popup (zdroj: https://addons.mozilla.org/en-US/firefox/addon/asteroids-in-popup/)**

Poslední skupinou jsou nástroje pro vývojáře (development tools). Lze tak pomoci vývojářům při tvorbě webových aplikací ladit chyby a navrhovat řešení i v prohlížeči. (Mozilla, 2018) Potencionálně je možno vylepšit stávající vestavěné nástroje prohlížeče přidáním nové karty do panelu nástrojů. Jako příklad lze zmínit React Developer Tools[15]. Tento doplněk umožnuje ladit a procházet React komponenty na webu, pokud je detekuje.

Existuje spousta těchto rozšíření, které vývojáři hojně používají. Z mé praxe mohu dále zmínit Redux DevTools[16], což je opět doplněk pro ladění webových stránek, které jsou napsané v JavaScriptu a pro uchovávání stavu aplikace používají knihovnu Redux. O té podrobněji v samostatné kapitole.

Doplněk, který je výsledkem mé práce, spadá právě do poslední zmíněné skupiny. Je to tedy nástroj pro vývojáře, který by jim měl pomoci s vývojem a odhalit chyby, pokud chtějí dodržovat konvenci BEM.

[15] Odkaz na doplněk https://chrome.google.com/webstore/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi
[16] Odkaz na doplněk https://github.com/zalmoxisus/redux-devtools-extension




**Obrázek 7: Ukázka devtools doplňku React developer tools (zdroj: https://github.com/zalmoxisus/redux-devtools-extension)**

### *3.3.3 Struktura web extensions*

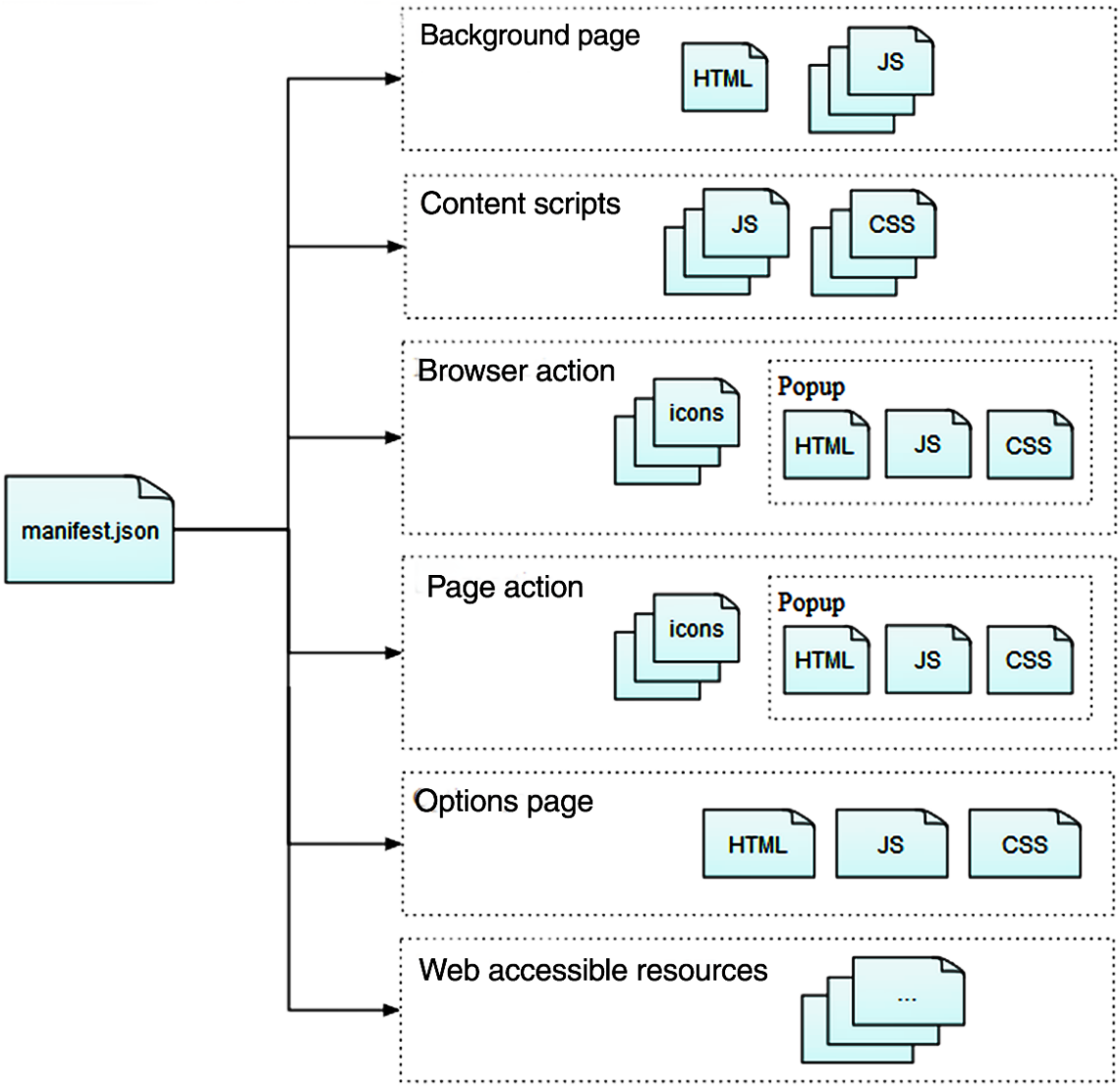
Každá aplikace typu web extension obsahuje sadu zabalených souborů pro distribuci a instalaci. Musí splňovat určité náležitosti. (Mozilla, 2018)

Především musí v kořenové složce obsahovat stěžejní soubor, kterému se říká manifest. Tento soubor musí být validním formátem typu JSON – pokud by tomu tak nebylo, nastane chyba při inicializaci aplikace. (Browser Extension Community Group , 2017)

Tento soubor je pojmenován jako *manifest.json* a obsahuje metadata jako jméno, verzi, klíčová nastavení a oprávnění aplikace. Manifest může obsahovat ukazatele na další soubory v doplňku.

- Skripty na pozadí.
- Ikony pro rozšíření a jakákoliv tlačítka, která by mohla definovat.
- Postranní panely (Sidebars), vyskakovací okna a stránky možností: dokumenty HTML, které poskytují obsah pro různé komponenty uživatelského rozhraní.
- Obsahové skripty: JavaScript, který je součástí rozšíření, který se bude vkládat do webových stránek.

- Zdroje přístupné webu: Zabezpečení, aby byl zabalený obsah přístupný webovým stránkám a obsahovým skriptům. (Mozilla, 2018)




**Obrázek 8: Struktura web extension (zdroj: Mozilla, 2018)**

Podrobněji v kapitole o manifestu 3.3.3.1.

Dále může složka obsahovat soubor pro stránku pozadí (*backround script*), který obvykle nese jméno background.js a jak již název souboru napovídá, jde o JavaScriptový soubor. Důležité je, jak je soubor namapován v rámci manifestu.

Tento soubor slouží k uchování dlouhodobého stavu nebo umožňuje provádět dlouhodobé operace nezávisle na životnosti určité webové stránky nebo okna prohlížeče. (Mozilla, 2018)

Skripty na pozadí se načítají, jakmile je rozšíření načteno a zůstanou načteny, dokud není rozšíření zakázáno nebo odinstalováno. Uvnitř skriptu lze využít libovolné rozhraní API WebExtension (v samostatné kapitole), pokud jsou udělena potřebná oprávnění. (Mozilla, 2018). Více o *backround script* v kapitole 3.3.5

Rozšíření může obsahovat různé komponenty uživatelského rozhraní, jejichž obsah je definován pomocí dokumentu HTML:

- postranní panel – je panel, který se zobrazuje na levé straně okna prohlížeče vedle webové stránky,
- rozbalovací okno – je dialog, který se může zobrazit, když uživatel klikne na tlačítko panelu nástrojů nebo tlačítko na panelu adres,
- stránka s volbami – je stránka, která se zobrazuje, když uživatel přistupuje k nastavení doplňku v manažeru doplňků prohlížeče. (Mozilla, 2018)

V případě použití těchto komponent se musí vytvořit soubor HTML a uvnitř něj použít konkrétní vlastnosti v *manifest.json* souboru. Zmíněný HTML soubor může obsahovat soubory CSS a JavaScript, stejně jako normální webová stránka.

Všechny tyto druhy komponent jsou typy rozšíření (*Extension pages*) a na rozdíl od běžné webové stránky v nich může JavaScript používat všechny stejné API rozhraní WebExtension jako *backround script*. Lze v nich přímo přistupovat k proměnným na pozadí stránky pomocí metody ***runtime.getBackgroundPage()***. (Mozilla, 2018)

Do rozšíření je možno zahrnout dokumenty HTML, které nejsou připojeny k předdefinované komponentě uživatelského rozhraní. Tyto soubory nemají v manifestu žádné záznamy, ale i tak mají přístup ke všem stejným privilegovaným API WebExtension jako skript na pozadí.

Obsahové skripty se používají pro přístup a manipulaci s webovými stránkami. Tyto skripty jsou načteny do webových stránek a spouštěny v kontextu této konkrétní stránky. Jsou poskytované rozšířením, které běží v kontextu webové stránky; liší se od skriptů, které jsou načteny samotnou stránkou, včetně těch, které jsou součástí elementů ***<script>*** v rámci stránky.

Obsahové skripty mohou manipulovat s DOMem, stejně jako normální skripty načítané stránkou.

Na rozdíl od běžných skriptů stránek mohou:

- provádět XHR požadavky mezi doménami,
- používat podmnožinu rozhraní *API WebExtension,*
- vyměňovat si zprávy se skripty na pozadí a tímto způsobem nepřímo přistupovat ke všem rozhraním *API WebExtension*.

Obsahové skripty tedy nemohou přímo přistupovat k běžným skriptům stránek, ale mohou si s nimi vyměňovat zprávy prostřednictvím standardního ***window.postMessage()*** API. Více v samostatné kapitole 3.3.5.1.

Na další zdroje jako obrázky, HTML, CSS a JavaScript, které jsou zahrnuty do rozšíření a mají být zpřístupněné obsahovým skriptům a skriptům webových stránek, lze odkazovat pomocí skriptů stránek a obsahových skriptů pomocí speciálního schématu URI.

Pokud chce například obsahový skript vložit některé obrázky do webových stránek, lze je přidat do rozšíření a zpřístupnit je webu. Pak obsahový skript může vytvořit a připojit *img tagy*, které odkazují na obrázky pomocí atributu ***src***. (Mozilla, 2018)

#### *3.3.3.1 Manifest*

Odpovídá souboru *manifest.json* a je to jediný soubor, který musí web extension obsahovat.

Soubor musí být formátu JSON, a může obsahovat komentáře typu **„//“**.

V následující tabulce se nachází přehled klíčů, které soubor může obsahovat. Zároveň tabulka obsahuje i vysvětlení a nezbytnost existence. (Mozilla, 2019)

| Klíč | Povinný | Podrobnosti |
|---|---|---|
| `"name"` | ano | Jméno rozšíření. |
| `"version"` | ano | Verze aplikace jako řetězec. |
| `"author"` | Ne | Jméno autora, které se zobrazuje v prohlížeči. |
| `"default_locale"` | Ano, pokud rozšíření používá překlady. Není povoleno, pokud je nepoužívá. | Více v kapitole o překladech a objektu ***browser.i18n*** 3.3.4.2.3. |
| `"description"` | ne | Popis rozšíření. |
| `"icons" : {`<br>`"<size>" : "<file-name.jpg>"`<br>`}` | ne | Seznam dvojic rozměrů v pixelech a názvů souborů PNG a SVG formátu (obrázků). Většinou jsou zobrazeny ve správci doplňků prohlížeče. Prohlížeče mohou vyžadovat konkrétní velikosti pro různá zobrazení.<br>`"icons":{`<br>`"16":"small.png",`<br>`"32":"medium.png"`<br>`}` |
| `"developer": {`<br>`"name": "<Company>",`<br>`"url": "<Website>"`<br>`}` | ne | Informace o vývojáři, konkrétně jde o jméno a odkaz na web vývojáře. Nejde o odkaz s podrobnostmi o rozšíření. |
| `"commands": {`<br>`<feature>: {..}`<br>`}` | ne | Nastavení klávesových zkratek pro rozšíření. |
| `"devtools_page": <path>` | ne | Povoluje rozšíření nástroje pro vývojáře. Obsahuje cestu k HTML souboru, který se tam bude zobrazovat. |
| `"browser_action"` | ne | Více v kapitole o ***browserAction*** 3.3.4.2.1. |
| `"page_action"` | ne | Více v kapitole o ***pageAction*** 3.3.4.2.4. |
| `"browser_specific_settings":`<br>`{`<br>`<browser_name>": {`<br>`" "<key>": "<value>"`<br>`}`<br>`}` | ne | Obsahuje specifické hodnoty pro různé prohlížeče. Je možno nastavit minimální podporovanou verzi prohlížeče pro běh rozšíření.<br>`"browser_specific_settings":{`<br>`"gecko": {`<br>`"strict_min_version": "42.0"`<br>`}`<br>`}`<br>Není podporováno v Google Chrome. |

| | | |
|---|---|---|
| `"background": { "page": "<Page URL>", "scripts": [...], "persistent": boolean, }` | ne<br><br>Pokud je definováno, je klíč persistent vyžadován. Ostatní podklíče jsou volitelné. | Automaticky načte stránku nebo skript a začne se provádět bez ohledu na stránku, která se zobrazuje na kartě. |
| `"content_scripts": { "all_frames": <boolean>, "css": [...], "exclude_matches": [...], "js": [...], "matches": [...], "run_at": <see_details>, }` | ne | Více v kapitole 3.3.3.1.1. |
| `"content_security_policy"` | ne | Více v kapitole 3.3.3.1.2. |
| `"options_page"` | ne | Stránka v rámci rozšíření, která umožňuje uživatelům měnit nastavení nebo možnosti rozšíření. Zastaralá metoda. |
| `"homepage_url"` | ne | URL adresa doplňku. |
| `"incognito": <mode>` | ne | Omezení pro anonymní režim. Lze zde zakázat přístup k *localStorage* a více chránit soukromí uživatele. |
| `"offline_enabled": <boolean>` | ne | Povolení off-line režimu. Podporuje pouze Google Chrome. |
| `"omnibox": {<keyword>: <value>}` | ne | Propojení s Omnibox[17] API na základě zadaného klíčového slova. |
| `"optional_permissions": […]` | ne | Pole dodatečných oprávnění, která mohou být aplikována až po instalaci doplňku. |
| `"options_ui": {…}` | ne | Nově používaný klíč pro stránku v rámci rozšíření, která umožňuje uživatelům měnit nastavení nebo možnosti rozšíření. Používá se místo ***options_page***. |
| `"manifest_version"` | ano | Hodnota musí být: 2 |

[17]O Omnibox API https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API/omnibox

| | | |
|---|---|---|
| `"web_accessible_resources":[...]` | ne | Zdroje jako obrázky, skripty a soubory, které mohou být přístupné internetovým stránkám. Více v kapitole. |
| `"externally_connectable": [ "ids": [...], "matches": [...], ]` | ne | Ostatní rozšíření nebo webové stránky mohou posílat zprávy do rozšíření na základě definice ID nebo URL. Není podporováno ve Firefoxu. |
| `"permissions" : { "activeTab", "contextMenus", "tabs", "webNavigation", "webRequestBlocking", "<url-pattern>", }` | ne | Více v kapitole Permissions 3.3.3.1.3. |
| `"protocol_handlers": [{…}]` | ne | Díky tomuto klíči lze obsluhovat různé webové protokoly. Takto je možno přiřadit k různým typům odkazů různá chování. |
| `"short_name": <name>` | ne | Reprezentuje zkrácené jméno rozšíření. Maximálně 12 znaků. |
| `"sidebar_action": { "default_icon": { "16": "button/geo-16.png", "32": "button/geo-32.png" }, "default_title": "My sidebar", "default_panel": "sidebar/sidebar.html", "open_at_install":true }` | ne | Umožnuje nastavit panel na levé straně prohlížeče. Lze definovat ikony, cestu k HTML souboru a další. |
| `"theme": { "images": {…}, "colors: {…}, }` | ne | Velmi populární klíč v manifestu. Umožnuje nastavit vzhled prohlížeče. Ať už jsou obrázky nebo barvy. |
| `"version_name": <version>` | ne | Klíč využívaný k aktualizacím doplňků a může sloužit i jako dodatečný popisek verze. |

**Tabulka 2: Seznam klíčů pro manifest soubor s popisem (zdroj: Mozilla, 2019)**

Další oprávnění pro Chrome na URL.[18]

[18] Oprávnění pro prohlížeč Google Chrome https://developer.chrome.com/extensions/manifest

Ukázka souboru *manifest.json*.

```
{
  "browser_specific_settings": {
    "gecko": {
      "id": "addon@example.com",
      "strict_min_version": "42.0"
    }
  },

  "background": {
    "scripts": ["jquery.js", "my-background.js"],
    "page": "my-background.html"
  },

  "browser_action": {
    "default_icon": {
      "19": "button/geo-19.png",
      "38": "button/geo-38.png"
    },
    "default_title": "Whereami?",
    "default_popup": "popup/geo.html"
  },

  "commands": {
    "toggle-feature": {
      "suggested_key": {
        "default": "Ctrl+Shift+Y",
        "linux": "Ctrl+Shift+U"
      },
      "description": "Send a 'toggle-feature' event"
    }
  },

  "content_security_policy": "script-src 'self' https://example.com;
object-src 'self'",

  "content_scripts": [
    {
      "exclude_matches": ["*://developer.mozilla.org/*"],
      "matches": ["*://*.mozilla.org/*"],
      "js": ["borderify.js"]
    }
  ],

  "default_locale": "en",

  "description": "...",

  "icons": {
    "48": "icon.png",
    "96": "icon@2x.png"
  },

  "manifest_version": 2,

  "name": "...",
```

```
  "page_action": {
    "default_icon": {
      "19": "button/geo-19.png",
      "38": "button/geo-38.png"
    },
    "default_title": "Whereami?",
    "default_popup": "popup/geo.html"
  },

  "permissions": ["webNavigation"],

  "version": "0.1",

  "web_accessible_resources": ["images/my-image.png"]
}
```

**Ukázka obsahu souboru (zdroj: Mozilla, 2019)**

Přehled s podporou jednotlivých prohlížečů daných klíčových slov na odkazu[19].

#### *3.3.3.1.1 Content scripts*

Automaticky přidává skripty (ze seznamu podklíčů „*js*“) nebo CSS (ze seznamu podklíčů „*cs*s“) na každou stránku v kartě prohlížeče.

Pokud je hodnota klíče ***„all_frames“*** nastavena na ***false*** (defaultní hodnota), tak se o tom, jaký obsah stránek bude upraven či ne, rozhoduje podle nastavených hodnot polí klíčů ***„matches“*** a ***„exclude_matches“.***

Validní hodnoty pro klíč ***„run_at“*** jsou ***„document_start“***, ***„document_end“*** a ***„document_idle“***.

V případě hodnoty ***„document_start“*** jsou skripty vloženy v okamžiku načítání DOMu. U hodnoty ***„document_end“*** se vloží a načtou, až když je DOM načten, nicméně stále se mohou načítat další zdroje jako obrázky, styly a JavaScript. Výchozí hodnotou je ***„document_idle“*** a ta vede k tomu, že jsou načteny všechny zdroje i DOM. (Google Chrome, 2018)

[19] Přehled klíčových slov https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Browser_compatibility_for_manifest.json

```
"content_scripts": [
    {
      "js": ["my-script.js"],
      "matches": ["https://example.org/"],
      "match_about_blank": true,
      "all_frames": true
    }
  ]
```

#### *3.3.3.1.2 Content Security Policy (CSP)*

Rozšíření mají ve výchozím nastavení několik zásad k zabezpečení obsahu. Výchozí zásada omezuje zdroje, ze kterých se mohou načítat prostředky ***<script>*** a ***<object>*** a zakazuje potenciálně nebezpečné praktiky, jako je například použití funkce **eval()**[20].

Hodnotu v manifest souboru lze nastavit těmito způsoby:

- Vložením ***„script-src“*** nebo ***„object-src“*** bude umožněno rozšíření načítat skripty nebo objekty mimo balíček prostřednictvím vložení jejich URL cesty.
- Vložením ***„script-src“*** s hashem skriptu je možno spouštět inline skripty.
- Nastavením ***„unsafe-eval“*** je povoleno vykonávání funkce ***eval()*** a podobné další funkce[21].

Povinné hodnoty jsou ***„script-src***“ nebo ***„object-src“***. První z nich musí obsahovat klíčové slovo ***„self“***. Zdroje, které jsou umístěny mimo balíček, musí odpovídat HTTPS schématu.

Jediné povolená klíčová slova jsou: ***„none“, „self“*** a ***„unsafe-eval“***. (Mozilla, 2018)

Příklady validního nastavení:

```
"content_security_policy": "script-src 'self' https://example.com;
object-src 'self'"
```

Povolí skripty z adresy https://example.com.

[20] Vysvětlení funkce na adrese https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/eval

[21] Ukázka podobných funkcí na URL (Mozilla, 2018)https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Content_Security_Policy#eval%28%29_and_friends

```
"content_security_policy": "script-src 'self' https://*.jquery.com;
object-src 'self'"
```

Povolí skripty ze subdomény jquery.com

```
"content_security_policy": "script-src 'self' 'unsafe-eval'; object-src
'self';"
```

Umožní volat ***eval()*** a podobné funkce.

```
"content_security_policy": "script-src 'self' 'sha256-
qznLcsROx4GACP2dm0UCKCzCG+HiZ1guq6ZZDob/Tng='; object-src 'self'"
```

Povolí inline skript „`<script>alert('Hello, world.');</script>`“

```
"content_security_policy": "script-src 'self'; object-src 'self'; img-src
'self'"
```

Umožní zabalit obrázky k rozšíření.

```
"content_security_policy": "default-src 'self'"
```

Všechny druhy obsahu musí být zabaleny k rozšíření. (Mozilla, 2018)

#### *3.3.3.1.3 Permissions*

Pro možnost využívání speciálních API prohlížeče je potřeba je povolit právě v manifestu.

```
"permissions": [
  "*://developer.mozilla.org/*",
  "webRequest"
]
```

Informace o tom, že je požadováno některé oprávnění, se objeví před instalací doplňku. Zde je potřeba pečlivě přemýšlet o požadovaných oprávněních, protože to může mít vliv na to, jestli se uživatel vůbec rozhodne instalovat doplněk.

Existují 3 skupiny oprávnění:

- povolení hostitele (webové stránky),
- povolení k používání API,
- povolení k používání aktivní karty prohlížeče (***activeTab***).

(Mozilla, 2019)

### *Povolení hostitele*

Povolení hostitele se zapisuje pomocí vzorů, které reprezentují skupinu URL adres, pro které rozšíření požaduje další oprávnění.

```
"*: //developer.mozilla.org/*".
```

Zvláštní oprávnění zahrnují:

- umožnění požadavku prostřednictvím *fetch* či *XMLHttpRequest* na tyto URL bez omezení cross-origin[22],

  povolení vkládat skripty z těchto URL zdrojů pomocí metody ***tabs.executeScript()*,**

- schopnost poslouchat na události ***webRequest***[23] API z těchto URL zdrojů,
- možnost přístupu k souborům ***cookies*** prostřednictvím cookies [24] API. (Mozilla, 2019)

### *Povolení k používání API*

K povolení používání API se deklaruje výčet jednotlivých klíčových slov, která budou používána a odpovídají názvům *WebExtension* API. (Mozilla, 2019)

V poslední verzi existují tyto klíčová slova:

| | | | |
|---|---|---|---|
| activeTab | alarms | background | bookmarks |
| browserSettings | browsingData | contentSettings | contextMenus |
| contextualIdentities | cookies | debugger | dns |
| downloads | downloads.open | find | geolocation |
| history | identity | idle | management |
| menus | nativeMessaging | notifications | pageCapture |

[22] O Cross-origin na adrese https://www.interval.cz/clanky/ajax-cors-polling/
[23] O WebRequest na URL https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API/webRequest
[24] O cookies na stránce https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API/cookies

| | | | |
|---|---|---|---|
| pkcs11 | privacy | proxy | search |
| sessions | storage | tabHide | tabs |
| theme | topSites | webNavigation | webRequest |
| webRequestBlocking | | | |

**Tabulka 3: Seznam klíčových slov WebExtension API (zdroj: Mozilla, 2019)**

Výjimky lze najít na stránkách dokumentace[25].

Přiklad zápisu:

```
"permissions": ["tabs"]
```

### *Povolení k používání aktivní karty prohlížeče*

Jak název odpovídá, jde o povolení práce s aktivní kartou prohlížeče. Takto lze zachytit následují události:

- Uživatel klepne na libovolnou akci rozšíření.
- Uživatel zvolí položku kontextové nabídky.
- Uživatel aktivuje klávesovou zkratku definovanou rozšířením.

Dále je k dispozici umožnění přístupu ke schránce (*clipboard*) [26], případně k neomezenému uložišti[27]. (Mozilla, 2019)

Podrobnosti na URL dokumentace[28].

[25] https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/manifest.json/permissions#API_permissions
[26] O přístupu ke stránce nahttps://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/manifest.json/permissions#Clipboard_access
[27] Informace k neomezenému uložišti na https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/manifest.json/permissions#Unlimited_storage
[28] Dokumentace Permissions https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/manifest.json/permissions

### 3.3.4 Specifikace API

Specifikace API obsahuje následující funkcionality:

- modifikace obsahu webové stránky,
- rozšíření UI webového prohlížeče,
- modifikace síťových požadavků,
- výměna dat mezi jednotlivými rozšířeními (*extensions*),
- výměna dat mezi webovými stránkami a rozšířeními,
- přístup uživatelským datům (např.: *cookies*).

(Browser Extension Community Group , 2017)

#### 3.3.4.1 Dostupnost API

Rozšíření do prohlížeče jsou vytvářena především v jazyce JavaScript. Kromě obvyklého *JavaScript Web API*[29] (tedy práce s DOMem[30]) jsou dostupná další rozhraní pro samotná rozšíření prostřednictvím browser objektu (více v kapitole 3.3.4.2). Konkrétní rozhraní ale ovlivňují:

- Kontext spouštění (např. pozadí/událost, vyskakovací okno, možnosti, obsah, stránka s rozšířením).
- Klíče deklarované v manifestu (např. ***browser_action, action_page***). Kapitola 3.3.4.2.
- Oprávnění deklarovaná v manifestu (např. ***ContextMenu, tabs, webNavigation, webRequest***).
- Politika zabezpečení obsahu (CSP) pro rozšíření a cílovou webovou stránku (pro obsahové skripty) (Browser Extension Community Group , 2017)

[29] Dokumentace Web API na https://developer.chrome.com/extensions/api_other
[30] O DOMu na URL http://www.adaptic.cz/znalosti/slovnicek/dom/

#### *3.3.4.1.1 Kontext spuštění a manifest restrikcí*

Stránky s rozšířením prohlížeče a jejich příslušné skripty jsou spuštěny v jednom z následujících kontextů. Definice klíčů v manifestu, které jsou potřebné k vytvoření každého konkrétního typu stránky, jsou popsány níže:

- Kontexty rozšíření prohlížeče ***(„window“***) – většina API rozšíření je v tomto kontextu k dispozici.
    - Stránka na pozadí (***Background page***) – používá se klíčová značka ***„background“*** se specifikací ***„president“*** na hodnotu ***true***.
    - Stránka události (***Event page***) – používá se klíčová značka ***„background“*** se specifikací ***„persistent“*** na hodnotu ***false***.
    - Vyskakovací stránka (***Popup page***) – používá klíč manifestu ***„browser_action“*** nebo ***„page_action“*** s hodnotou ***„default_popup“***.
    - Stránka možností (***Options page***) – použije atribut manifestu ***„options_page***“.
    - Vlastní stránka (***Custom page***) – například *„browserext: // <ext_id> /myCustomPage.html“*.
- Obsah kontextu skriptu (***Content script context***) – k dispozici je omezený počet rozhraní API, která skripty rozšíření používají k interakci s hostovanými webovými stránkami.
- Kontext webových stránek hostovaných na internetu (***internet-hosted web page context***) – ***Messaging API*** může být použito k odesílání zpráv z hostovaných webových stránek (např. *https://en.wikipedia.org/wiki/Main_Page/index.html*) k rozšířením.

(Browser Extension Community Group , 2017)

Některé API pro rozšíření prohlížeče vyžadují specifický klíč nebo konkrétní oprávnění v manifestu. Pokud se tak nestane, objekt nebude k dispozici pro další práci a nebude ho možno využít ke skriptování.

Tyto požadavky jsou shrnuty v následující tabulce. Pokud jsou zadány manifestní klíče, které nejsou definovány v této specifikaci, mohou je ostatní prohlížeče ignorovat. (Browser Extension Community Group , 2017)

| Objekt API | Kontext spuštění | | | Požadovaná deklarace |
|---|---|---|---|---|
| | **Background page, Event page, Popup page, Options page, Custom page** | **Content script page** | **Internet-hosted web page context** | **Klíč manifestu** |
| ***browserAction*** | Ano[1] | | | "browser_action"[1] |
| ***contextMenus*** | Ano | | | "permissions": ["context_menus"] |
| ***extension*** | Ano | Ano | | |
| ***i18n*** | Ano | Ano | | "default_lang": "<lang>"[4] |
| ***pageAction*** | Ano[1] | | | "page_action"[1] |
| ***runtime*** | Ano | Ano | Ano[3] | "externally_connectable": [...][2] |
| ***tabs*** | Ano | | | "permissions": ["tabs","activeTab"] |
| ***webNavigation*** | Ano | | | "permissions": ["webNavigation"] |
| ***webRequest*** | Ano | | | "permissions": ["webRequest"] |
| ***windows*** | Ano | | | |

**Tabulka 4: Požadavky na klíče v manifestu pro objekty API (zdroj: Browser Extension Community Group , 2017)**

1 Lze povolit ***browserAction*** nebo ***pageAction***, oboje najednou nelze

2 Runtime objekt je k dispozici pro rozšíření bez deklarace ***„externally_connectable“*** v manifestu. Rozšíření však přijme pouze zprávy odeslané z hostovaných webových stránek, pokud je tento klíč deklarován. Deklarace ***„externally_connectable“*** zablokuje zprávy z jiných rozšíření, pokud není v seznamu "IDS" deklarováno povolené ID rozšíření.

3 Webové stránky hostované na internetu, které používají ***runtime.sendMessage*** pro komunikaci s rozšířeními, musí být výslovně určeny doménu hostitelského webu v manifestu pod klávesou ***„exterally_connectable“.***

4 Použití objektu ***browser.i18n*** nevyžaduje deklaraci zřejmého oprávnění. Protokol však vyžaduje hodnotu (***default_locale***), která je nezbytná a nemůže jej zadat autor kódu v kódu skriptu. (Browser Extension Community Group , 2017)

#### 3.3.4.2 Browser objekt

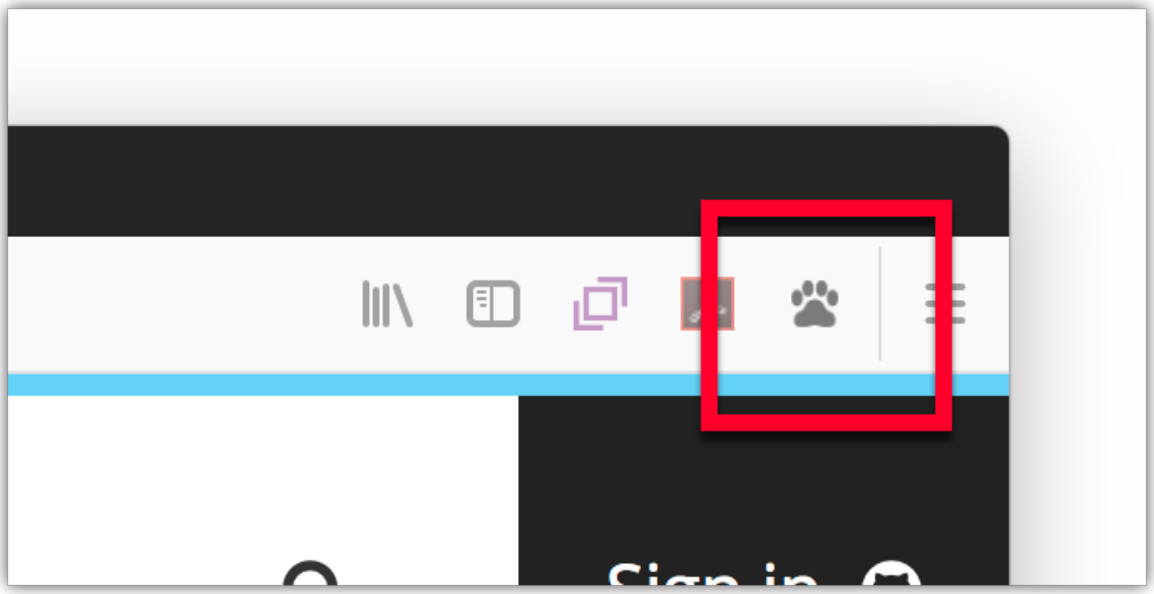
Základní objekty API pro rozšíření prohlížeče jsou přístupné z objektu ***browser***. Dostupnost těchto objektů se liší v závislosti na podmínkách uvedených v minulé kapitole. (Browser Extension Community Group , 2017)

##### 3.3.4.2.1 browserAction

Objekt přidává do aplikace prohlížeče vždy viditelné tlačítko, obvykle v horní části uživatelského rozhraní aplikace prohlížeče v oblasti panelu nástrojů. Rozšíření, které obsahuje ***browserAction***, nemůže obsahovat ***pageAction***, a naopak.

```
"browser_action": {
    "default_icon": {...},
    "default_popup": "Page URL",
    "default_title": "Title string"
}
```

Povinnou položkou je ***default_title***. (Browser Extension Community Group , 2017)



**Obrázek 9: Doplněk s Browser action definicí (zdroj: vlastní)**

##### 3.3.4.2.2 contextMenus

Objekt ***contextMenus*** umožňuje přidat další položky do kontextového menu prohlížeče. Nabídka bude selektivně zobrazena pro různé prvky uživatelského rozhraní prohlížeče (akce prohlížeče nebo tlačítko stránky) či prvky stránky (rámeček, obrázek, odkaz, stránka, výběr atd.).

Hodnota „***contextMenus***“ musí být deklarována v poli „***permissions***“ v souboru *manifest.json*. Atribut „***icons***“ je volitelný. (Browser Extension Community Group , 2017)

```
"permissions": [
    "contextMenus"
],
"icons": {
    "<size>":"<name.png>"
```

#### *3.3.4.2.3 i18n*

Objekt ***i18n*** poskytuje přístup k lokalizaci řetězců, které jsou přeloženy do podporovaných jazyků.

Každá podporovaná lokalizace musí mít svůj vlastní soubor ***messages.json***, který bude uložen ve složce ***_locales / <kodJazyka>***. Název adresáře odpovídá dvouznakovým kódům, které jsou stejné jako v manifestu rozšíření. Lze pak tedy specifikovat defaultní jazyk (***default_locale***). Kódy jazyků odpovídají normám RFC1766[31] nebo IETF BCP47[32]. Například *en_US* nebo *cs_CZ*.

```
"default_locale" : "en",
"name" : "__MSG_MySampleExtension__",
"description" : "__MSG_MySampleExensionDecription__",
```

Obsah souboru může vypadat následovně, kdy povinné jsou položky pro samotnou identifikaci („***name***“) a „***message***“ jako samotný obsah překladu. Přidat se dá i popis jako „***description***“. (Browser Extension Community Group , 2017)

```
{
  "extensionName": {
    "message": "Notify link clicks i18n",
    "description": "Name of the extension."
  },

  "extensionDescription": {
    "message": "Shows a notification when the user clicks on links.",
    "description": "Description of the extension."
  },

  "notificationTitle": {
    "message": "Click notification",
    "description": "Title of the click notification."
  },

  "notificationContent": {
    "message": "You clicked $URL$.",
    "description": "Tells the user which link they clicked.",
```

[31] https://www.w3.org/TR/1999/REC-html401-19991224/struct/dirlang.html
[32] https://www.w3.org/International/articles/language-tags/

```
    "placeholders": {
      "url" : {
        "content" : "$1",
        "example" : "https://developer.mozilla.org"
      }
    }
  }
}
```

**Ukázka zápisu v souboru[33]**

Jazykové zprávy v daném formátu jsou nahrazovány automaticky příslušným přeloženým řetězcem pomocí regulárního výrazu[34] „*_MSG_([A-Za-z0-9@_]+?)__“*.

V aplikaci se pak používají tzv. *placeholdery* (zástupné symboly), což jsou zástupné znaky překladu a je vhodné používat, pokud přeložený řetězec obsahuje podřetězce, které nebudou přeloženy nebo budou nahrazeny variabilním obsahem. Například:

```
"messageExample": {
    "message": "You clicked $URL$.",
    "description": "Tells the user which link they clicked.",
    "placeholders": {
      "url" : {
        "content" : "$1"
        "example" : "https://www.w3.org"
      }
```

Název zástupného symbolu („*url“* ve výše uvedeném příkladu) se používá k zastupování *placeholderu* v substitučním řetězci (např. „*U*RL“ se stává ***$ URL $***). Může obsahovat velká i malá písmena (case insensitive) a stejné znaky jako název řetězce zprávy.

Položka „*content*“ definuje obsah zástupného symbolu. Může to být řetězec, který je nastaven pevně (například „Můj placeholder“), ale může obsahovat i hodnoty získané z volání metody ***i18n.getMessage()***.

Nepovinná položka „*example*“ je určena k tomu, aby pomohla překladatelům ukázat jim příklad toho, kde a jak se může objevit zástupný symbol, a umožnit jim tak nejlepší výběr při lokalizaci konkrétního překladu. (Browser Extension Community Group , 2017)

Podrobnosti na webové stránce dokumentace[35].

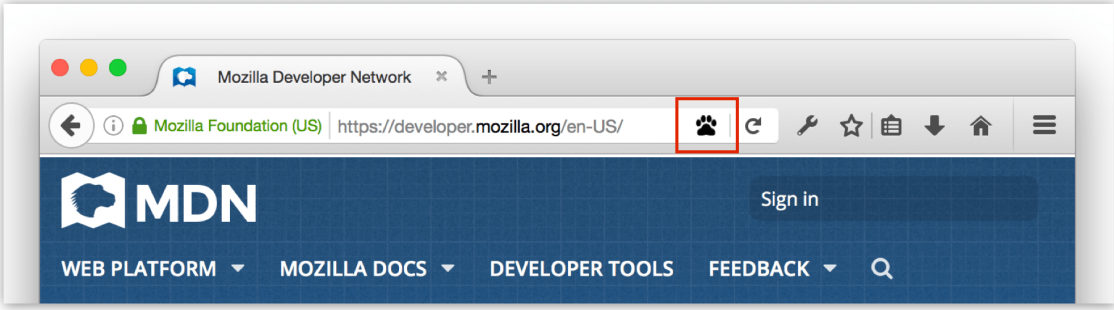
[33] https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API/i18n/Locale-Specific_Message_reference
[34] Regulární výrazy na https://www.regularnivyrazy.info/regularni-vyrazy-zaklady.html
[35] https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Internationalization

#### 3.3.4.2.4 pageAction

Objekt ***pageAction*** přidává tlačítko do aplikace prohlížeče, obvykle v adresním řádku. Toto tlačítko se obvykle zobrazí pouze v případě, že rozšíření zjistilo, že může provést akci na aktuální stránce aktivní karty. Rozšíření, které obsahuje ***browserAction*** nemůže obsahovat ***pageAction***, jak již bylo zmíněno výše. (Browser Extension Community Group , 2017)

```
"page_action": {
    "default_icon": {...},
    "default_popup": "Page URL",
    "default_title": "Title string"
}
```

Povinnou položkou je ***default_title***.




**Obrázek 10: Doplněk s nastaveným PageAction klíčem (zdroj: vlastní)**

#### 3.3.4.2.5 runtime

Tento objekt je možno použít k odesílání a přijímání zpráv v kontextech, stejně jako k přístupu na úrovni rozšíření, jako je id, manifest a absolutní cesta ke zdrojům.

Možnosti komunikace objektu:

- komunikace mezi částmi rozšíření,
- komunikace s cizími rozšířeními,
- komunikace s nativními aplikacemi.

```
"externally_connectable" : [
    "ids":[...],
    "matches":[...]
    ]
```

Pokud je v manifestu deklarován klíč „***externally_connectable***“, rozšíření zablokuje přijímaní a odesílání zpráv v závislosti na nastavení. (Browser Extension Community Group , 2017)

#### *3.3.4.2.6 tabs*

Pomocí tohoto objektu lze přistupovat k metadatům karet prohlížeče (název nebo obsah). Ať už k aktivní kartě (***activeTab***) nebo ke všem (***tabs***). Jak již bylo zmíněno výše, je zapotřebí přidat tuto hodnotu pro klíč „***permissions***“ v manifestu. (Browser Extension Community Group , 2017)

#### *3.3.4.2.7 webNavigation*

Díky objektu ***webNavigation*** je umožněno monitorovat události související se síťovými požadavky. Obdobně je nutné udělit přístup v manifestu. (Browser Extension Community Group , 2017)

#### *3.3.4.2.8 webRequest*

Zmíněný objekt umožňuje upravovat či zrušit síťový požadavek. Také se musí definovat jako hodnota pro klíčové slovo „***permissions***“. (Browser Extension Community Group , 2017)

#### *3.3.4.2.9 windows*

***Windows*** objekt slouží k přístupu uživatelského rozhraní prohlížeče. (Browser Extension Community Group , 2017)

### *3.3.5 Komunikace mezi vrstvami*

#### *3.3.5.1 Komunikace se skriptem na pozadí*

Jednotlivé vrstvy mezi sebou komunikují pomocí zpráv, odchytávají je jako události. Existují dvě možnosti, jak komunikovat, a to jednoduchými jednorázovými požadavky s nepovinnou odpovědí nebo trvalými spojeními. (Google Chrome, 2018)

#### *3.3.5.1.1 Jednorázové požadavky (one-time requests)*

Jednorázové požadavky se používají v případech, kdy je potřeba poslat jednu zprávu do jiné části rozšíření. Volitelně lze tímto způsobem ihned získat odpověď. Toto lze realizoval prostřednictvím API ***runtime.sendMessage*** nebo ***tabs.sendMessage***. Slouží k zaslání serializováné zprávy ve formátu JSON z obsahového skriptu do rozšíření a naopak. Druhý nepovinný parametr slouží ke zpracování odpovědi. (Google Chrome, 2018)

| | Obsahový skripty | Skript na pozadí |
|---|---|---|
| Poslání zprávy | ***browser.tabs.sendMessage()*** | ***browser.runtime.sendMessage()*** |
| Přijmutí zprávy | ***browser.runtime.onMessage()*** | ***browser.runtime.onMessage()*** |

**Tabulka 5: Metody pro posílání a přijímání zpráv pro různý kontext (zdroj: Mozilla, 2019)**

```
window.addEventListener("click", notifyExtension);

function notifyExtension(e) {
  if (e.target.tagName != "A") {
    return;
  }
  browser.runtime.sendMessage({"url": e.target.href});
}
// background-script.js

browser.runtime.onMessage.addListener(notify);

function notify(message) {
  browser.notifications.create({
    "type": "basic",
    "iconUrl": browser.extension.getURL("link.png"),
    "title": "You clicked a link!",
    "message": message.url
  });
}
```

**Ukázka zápisu kódu při komunikaci (Mozilla, 2019)**

#### *3.3.5.1.2 Trvalé spojení*

Trvalé spojení se doporučuje použít v případě, že se vyměňuje větší množství zpráv mezi vrstvami (obsahovým skriptem a skriptem na pozadí). Jde o alternativní přístup, kdy se vytvoří spojení mezi těmito vrstvami, které umožní vyměňovat zprávy.

Na každé straně je potřeba vytvořit objekt ***runtime.Port***, který lze získat vytvořením spojení, a to následujícími kroky:

- Jedna ze stran naslouchá spojení pomocí ***runtime.onConnect***.
- Druhá strana volá jednu z funkcí níže a vrátí zmiňovaný ***runtime.Port*** objekt.
    - ***tabs.connect()*** - při připojování z obsahového skriptu.
    - ***runtime.connect()*** – při připojování ze skriptu na pozadí.
- Listener ***runtime.onConnect*** předá svůj Port objekt.
- Každá ze stran vlastní svůj port, který slouží pro výměnu zpráv voláním metod.
    - ***runtime.Port.postMessage()*** – poslání zprávy.
    - ***runtime.Port.onMessage*** – přijmutí zprávy.

Příklad spojení v obsahovém skriptu:

1. Připojení ke skriptu na pozadí a uložené objektu port do proměnné.
2. Naslouchání na zprávy v proměnné ***myPort*** a jejich zaznamenávání.
3. V případě kliknutí uživatele v těle dokumentu zasílání zpráv do skriptu na pozadí.

```
// content-script.js

var myPort = browser.runtime.connect({name:"port-from-cs"});
myPort.postMessage({greeting: "hello from content script"});

myPort.onMessage.addListener(function(m) {
  console.log("In content script, received message from background
script: ");
  console.log(m.greeting);
});

document.body.addEventListener("click", function() {
  myPort.postMessage({greeting: "they clicked the page!"});
});
```

Korespondující kód pro skript na pozadí s posloupností kroků:

1. Naslouchá na pokusy připojení z obsahové skriptu a pokud se tak stane.
    a. Uloží si do objekt port do lokální proměnné.
    b. Odešle zprávu do obsahové skriptu.

c. Začíná naslouchat přijaté zprávy na portu.

2. Posílá do obsahového skriptu zprávy o tom, že uživatel kliknul na akci rozšíření (***browserAction***).

```
// background-script.js

var portFromCS;
function connected(p) {
  portFromCS = p;
  portFromCS.postMessage({greeting: "hi there content script!"});
  portFromCS.onMessage.addListener(function(m) {
    console.log("In background script, received message from content
script")
    console.log(m.greeting);
  });
}


browser.runtime.onConnect.addListener(connected);

browser.browserAction.onClicked.addListener(function() {
  portFromCS.postMessage({greeting: "they clicked the button!"});
});
```

Je dovoleno komunikovat s více obsahovými skripty, a to díky ukládání portů do pole. (Mozilla, 2019)

```
// background-script.js

var ports = []

function connected(p) {
  ports[p.sender.tab.id]    = p
  //...
}

browser.runtime.onConnect.addListener(connected)

browser.browserAction.onClicked.addListener(function() {
  ports.forEach(p => {
        p.postMessage({greeting: "they clicked the button!"})
    })
});
```

#### *3.3.5.2 Komunikace s webovou stránkou*

Komunikace s webovou stránkou je umožněna přes objekt ***window***, které nabízí metody ***window.postMessage*** a ***window.addEventListener***. První z nich slouží k posílání zpráv a druhá k jejich naslouchání. Přijme ***event*** objekt, který obsahuje atribut ***source*** (identifikátor zdroje), a data (samotná data k výměně). (Mozilla, 2019)

```
// page-script.js

var messenger = document.getElementById("from-page-script");

messenger.addEventListener("click", messageContentScript);

function messageContentScript() {
  window.postMessage({
    direction: "from-page-script",
    message: "Message from the page"
  }, "*");


// content-script.js

window.addEventListener("message", function(event) {
  if (event.source == window &&
      event.data &&
      event.data.direction == "from-page-script") {
    alert("Content script received message: \"" + event.data.message +
"\"");
  }
});
```

Uvnitř obsahového skriptu lze vykonávat příkazy a to pomocí funkce ***eval()***, avšak jen v jeho kontextu. (Mozilla, 2019)

### 3.3.6 Práce se soubory

Existuje několik mechanismů, jak pracovat se soubory v rámci rozšíření:

- Stahování souborů do vybrané složky uživatele.
- Otevření souborů přes formulářové pole na webové stránce.
- Otevírání souborů pomocí *drag & drop*[36] na webové stránce.
- Uložení souborů lokálně do *IndexedDB*[37].
- Přenášení souborů do nativní aplikace v počítači uživatele.

(Mozilla, 2017)

### 3.3.7 Devtools rozšíření

Jelikož se tato závěrečná práce zabývá vytvořením právě tohoto druhu rozšíření, je nezbytné popsat i základní informace pro jeho vytvoření.

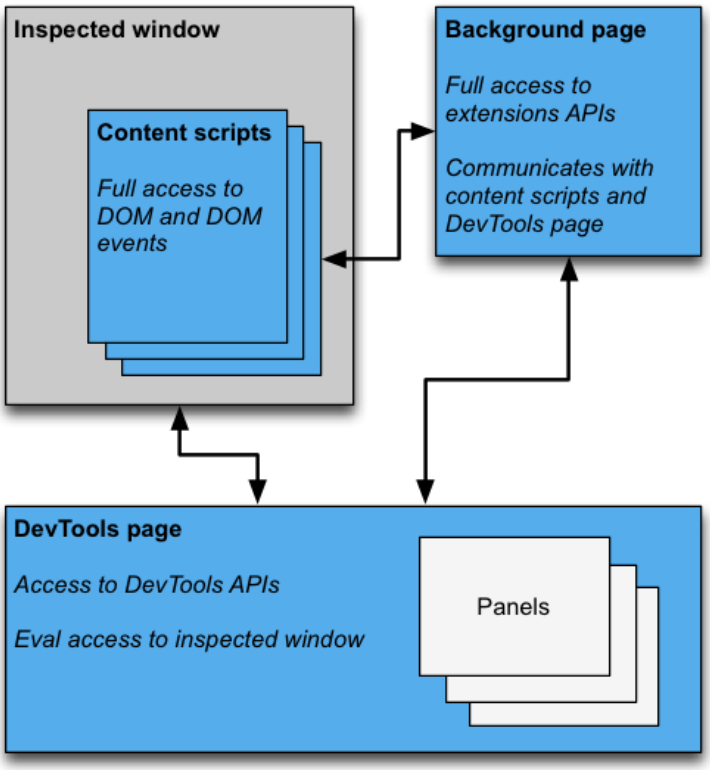
[36] Podrobnosti k drag & drop https://www.w3schools.com/html/html5_draganddrop.asp
[37] O IndexedDB https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API

#### 3.3.7.1 Základní charakteristika

Tento druh rozšíření přidává panel do nástrojů vývojáře v prohlížeči, dále může komunikovat s webovou stránkou, získávat informace o síťových požadavcích atd. Toto rozšíření má přístup ke speciálnímu API:

- ***devtools.inspectedWindow*** [38] - komunikace s oknem, pro které je otevřeno rozšíření.
- ***devtools.network***[39] - přístup k síťovým požadavkům.
- ***devtools.panels***[40] – operace s panely v nástrojích pro vývojáře.

Struktura tohoto druhu rozšíření je stejná jako u ostatních, jen navíc obsahuje i stránku devtools.html s přístupem k výše zmíněnému API. (Google Chrome, 2018)




**Obrázek 11: Schéma komunikace mezi částmi aplikace (zdroj: https://developer.chrome.com/extensions/devtools)**

[38] Podrobnosti o API https://developer.chrome.com/extensions/devtools_inspectedWindow
[39] Podrobnosti o API https://developer.chrome.com/extensions/devtools_network
[40] Podrobnosti o API https://developer.chrome.com/extensions/devtools_panels

Instance rozšíření vznikne při otevření Devtools panelu a existuje stejně dlouho jako samotná stránka.

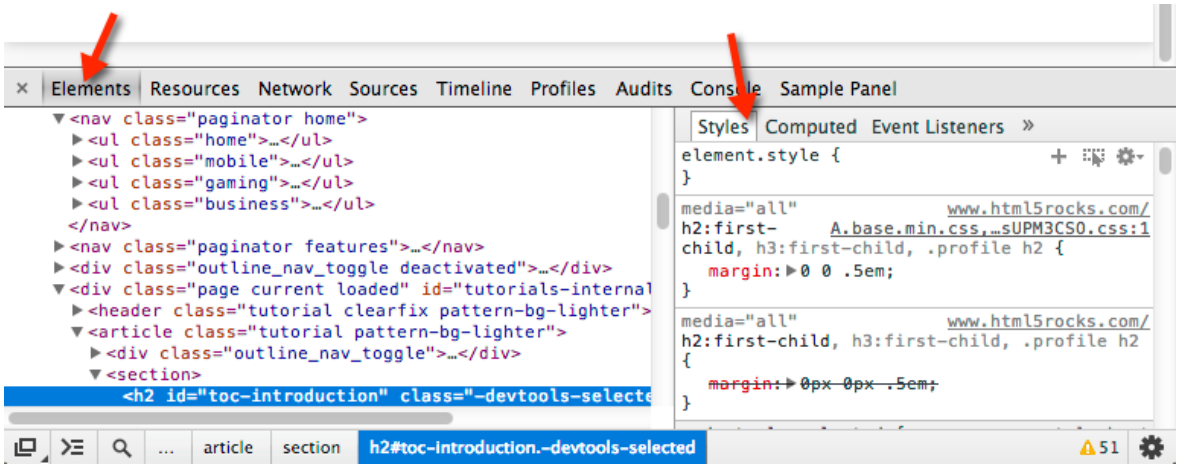
#### 3.3.7.2 Postup vytvoření rozšíření

Postup pro vytvoření devtools rozšíření se skládá z následujících kroků:

1. Prvním krokem pro vytvoření takového rozšíření je definice *manifest.json* souboru s požadovaným nastavením a propojením s devtools.html stránkou.

```
{
  "name": ...
  "version": "1.0",
  "devtools_page": "devtools.html",
  ...
}
```

2. Dalším krokem je vytvoření panelu pomocí ***devtools.panels*** API. Takto lze vytvořit panel na vyšší úrovni nebo postranní panel.




**Obrázek 12: Panel pro vývojáře (zdroj: vlastní)**

3. Poté je zapotřebí navázat spojení mezi jednotlivými komponentami, jak bylo zmíněno výše. Zde je zapotřebí při posílání zpráv identifkovat i kartu prohlížeče v případě, že se posílá zpráva do skriptu na pozadí.

```
// DevTools page -- devtools.js
// Create a connection to the background page
var backgroundPageConnection = browser.runtime.connect({
    name: "devtools-page"
});

backgroundPageConnection.onMessage.addListener(function (message) {
    // Handle responses from the background page, if any
});
```

```
browser.runtime.sendMessage({
    tabId: browser.devtools.inspectedWindow.tabId,
    scriptToInject: "content_script.js"
});

// Background page -- background.js
browser.runtime.onConnect.addListener(function(devToolsConnection) {
    var devToolsListener = function(message, sender, sendResponse) {
        browser.tabs.executeScript(message.tabId,
            { file: message.scriptToInject });
    }

    devToolsConnection.onMessage.addListener(devToolsListener);

    devToolsConnection.onDisconnect.addListener(function() {
         devToolsConnection.onMessage.removeListener(devToolsListener);
    });
}
```

4. Pro případ, že se má vykonávat nějaký příkaz v kontextu okna, pro které je otevřen panel pro vývojáře, je třeba využít API ***inspectedWindow.eval***.

```
browser.devtools.inspectedWindow.eval(
      "inspect($$('head script[data-soak=main]')[0])",
      function(result, isException) { }
    );
```

5. Pro výměnu zpráv mezi obsahovými skripty a devtools se používá API ***tabs.sendMessage***.
6. Pro komunikaci z vložených skriptů do devtools lze použít ***window.postMessag***e API.
7. V poslední řadě dává devtools informaci o tom, v jakém je stavu, třeba při otevření a zavření nebo načítání.

(Google Chrome, 2018)

### *3.3.8 Podpora JavaScript API v prohlížeči*

Prohlížeče bohužel nepodporují kompletní JavaScript API. Přehled lze najít zde[41]. Je velmi důležité si to uvědomit, protože pokud chce vývojář psát moderní JavaScript, musí buď použít transformaci (*Babel*) nebo pracovat jen s podporovaným API. (Mozilla, 2019)

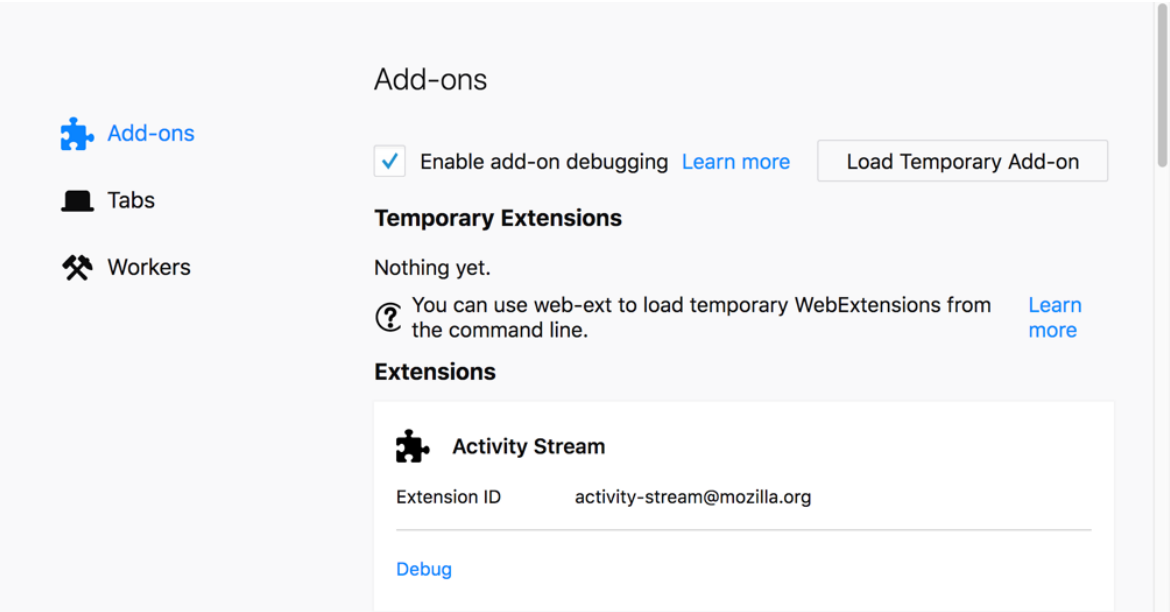
[41] Podporované API prohlížečů https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Browser_support_for_JavaScript_APIs

### 3.3.9 Testování doplňku během vývoje

Pro testování během vývoje rozšíření je potřeba aplikaci nainstalovat. Obvykle se prohlížeč nabízí speciální URL, kde se zobrazí možnosti pro nahrání a instalaci rozpracované verze doplňku

#### 3.3.9.1 Firefox

V prohlížeči Firefox jde o URL ***about:debugging***, která nabídne speciální formulář pro nahrání a ladění doplňků. Pro ladění se otevře speciální okno prohlížeče s konzolí, kde lze pozorovat případné chyby, či krokovat běh programu. (Mozilla, 2019)

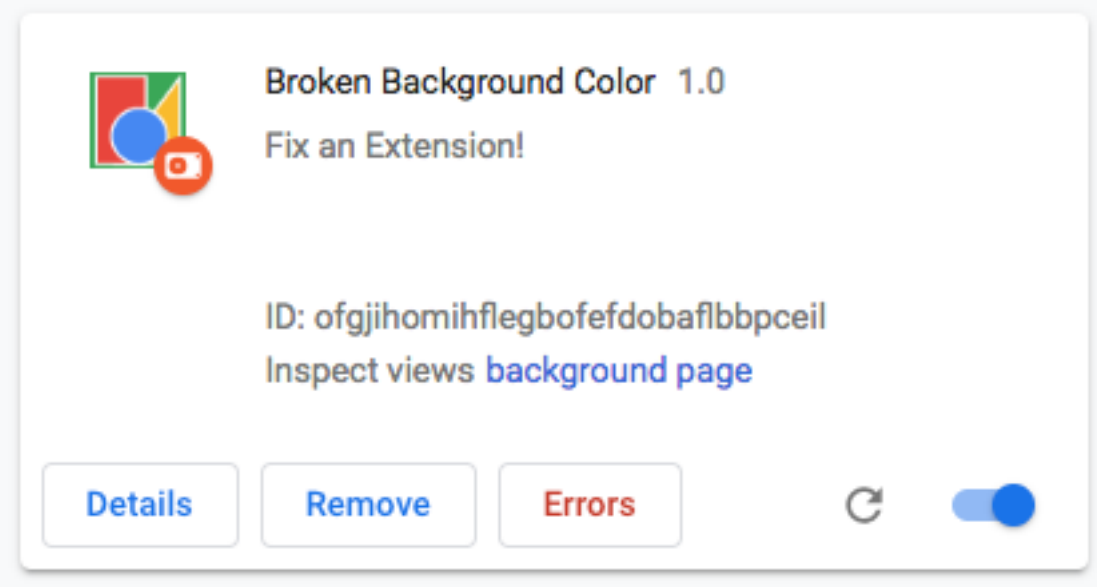



**Obrázek 13: Stránka pro správu doplňků v prohlížeči Firefox (zdroj: vlastní)**

#### 3.3.9.2 Chrome

Ve zmiňovaném prohlížeči Chrome je obdobný formulář dostupný na adrese ***chrome://extensions***, kde po zapnutí volby pro vývojáře umožní prohlížeč nainstalovat nezabalenou složku se soubory doplňku.

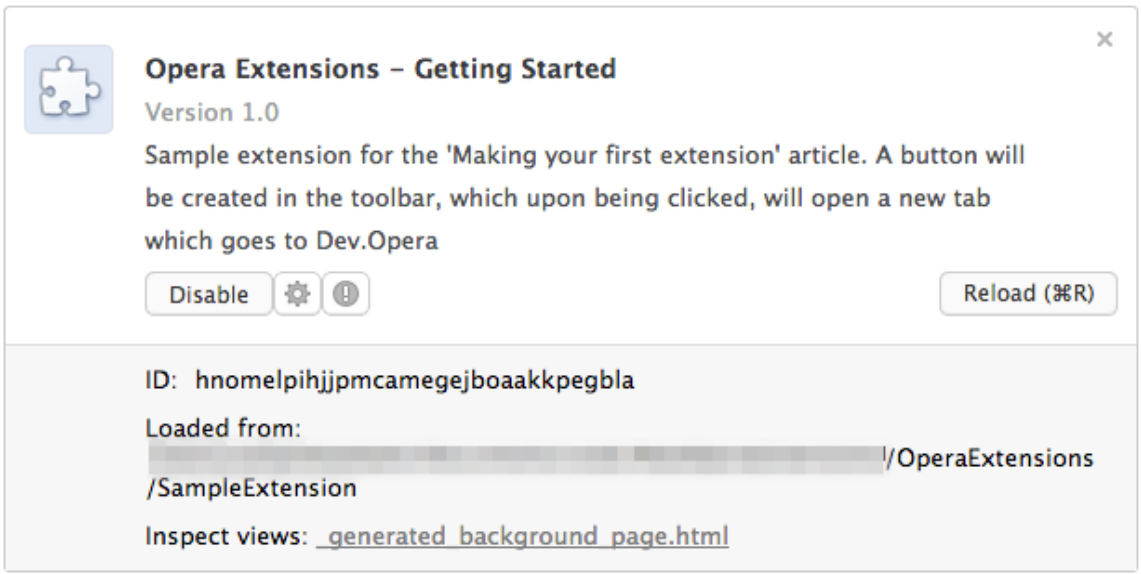
Rovněž tu je odkaz pro ladění a možnost prohlédnout si chyby, které vznikly během instalace, či spuštění doplňku. (Google Chrome, 2018)




**Obrázek 14: Detail doplňku ve správě doplňků v prohlížeči Chrome (zdroj: https://developer.chrome.com/extensions/tut_debugging)**

#### *3.3.9.3 Opera*

V případě prohlížeče Opera je to postup obdobný jako v Chrome. Formulář je na URL adrese ***opera://extensions***. (Dev.Opera, 2019)




**Obrázek 15: Detail doplňku ve správě doplňků v prohlížeči Opera (zdroj: https://dev.opera.com/extensions/testing/)**

#### 3.3.9.4 Ostatní prohlížeče

Podobným způsobem lze doplněk nainstalovat a ladit i v Microsoft EDGE, Opera, Vivaldi. Chromium či Brave. Podrobnosti na URL[42].

#### 3.3.9.5 Další nástroje pro ladění a testování

##### 3.3.9.5.1 Web-ext

Open source nástroj *Web-ext*[43] dovoluje vývojářům nainstalovat funkcionalitu do CLI. Mezi hlavní funkce nástroje patří spuštění doplňku přímo z příkazové řádky, lintování kódu, buildování pro produkční nasazení a podpis balíčku. Podepsání balíčku je potřeba pro jeho další distribuci v repozitářích prohlížečů. Nástroj lze nainstalovat z NPM.

Důležitá je zejména statická kontrola kódu (linting), kde je potřeba opravit všechny chyby, aby pak bylo možno rozšíření dále publikovat.

Příkaz build vytvoří zabalený soubor pro pozdější publikaci. (Mozilla, 2019)

##### 3.3.9.5.2 ESLint

Nástroj pro vývojáře, který je dostupný na NPM a pomáhá vývojáři udržovat kvalitu jeho kódu podle různých standardů. Nástroj lze integrovat do IDE nebo spouštět v CLI. Pro nástroj existuje řada předdefinovaných presetů, parserů a pluginů. Povolit se dá i mód pro tvorbu web extensions[44]. (JS Foundation, 2019)

##### 3.3.9.5.3 Stylelint

Podobný nástroj jako předchozí, jen s tím rozdílem, že pomáhá udržovat zápis CSS v preprocesorech LESS nebo SASS (SCSS). Pravidla si lze definovat i vlastní, nicméně existuje spousta presetů, které lze použít. (Michálek, 2018)

[42] Podrobnosti k instalaci a ladění doplňků v prohlížečích https://www.smashingmagazine.com/2017/04/browser-extension-edge-chrome-firefox-opera-brave-vivaldi/
[43] Domovská stránka balíčku Web-ext https://github.com/mozilla/web-ext
[44] Více o módu ESLint pro web extensions https://github.com/mdn/webextensions-examples/tree/master/eslint-example

#### 3.3.9.6 Jednotkové testy

Nedílnou součástí pro psaní spolehlivého a kvalitního kódu je pokrytí nejdůležitější logiky unit testy.

Pro tuto úlohu existuje spousta nástrojů. Jedním z nejvíce používaných je Jest od společnosti Facebook. Jest dovoluje v kombinaci s několika dalšími nástroji velmi rychle a efektivně testovat jednotlivé části kódu.

Speciálně se hodí na testování React komponent, kde využívá tzv. snapshot testů, kdy si uloží do souboru otisk komponenty jako vzorek, se kterým pak porovnává další běhy testu. Formát vzorku může být různý, nejčastěji se používá JSON. (Gimeno, 2018)

### 3.3.10 Distribuce a publikace doplňku

V případě, že je balíček s rozšířením hotov a otestovaný, lze přejít k samotné publikaci v jednotlivých repozitářích prohlížečů.

#### 3.3.10.1 Firefox

Oficiální stránkou Mozilly pro publikaci doplňků je adresa ***addons.mozilla.org*** (AMO). Zde vývojáři publikují své doplňky a uživatelé si je zde mohou stahovat do prohlížečů.

K publikaci je nutno mít balíček doplňku podepsaný společností Mozilla. Také je potřeba, aby manifest soubor obsahoval unikátní id doplňku.

Rozšíření lze instalovat ze souboru nebo jako standardní formát XPI.

K podepsaní je zapotřebí vlastnit vývojářský účet na portálu a pak lze využít k podpisu nástroje Web-ext.

Stejným nástrojem lze balíček i vytvořit a nahrát na web. Zde ho pak může vývojář verzovat, updatovat a také potvrdit jako produkční. (Mozilla, 2018)

Z výše zmíněného důvodu je zapotřebí správně uvádět verze do souboru *manifest.json*. V případě, že chce vývojář umožnit automatické aktualizace rozšíření, je nutné přidat do manifestu klíč ***update_url*** s hodnotou. (Mozilla, 2018)

Více informací na stránkách dokumentace[45].

#### 3.3.10.2 Chrome

Pro Chrome existuje samostatný store a publikace je obdobně složitá, navíc vývojáři musí platit poplatek. Pro publikace je potřeba mít vytvořen Google účet a zaregistrovat se jako vývojář. Dále pak nahrát aplikaci, zaplatit poplatek 5$ jednorázově za členství a po kontrole od Google získat ověřovací ID a token pro přidání do aplikace. Postup publikace lze najít na webu[46].

Výhodou je, že lze nastavit viditelnost publikace balíčku, kdy je možno vybrat z voleb veřejná publikace, soukromá publikace nebo publikace pro seznam testerů. (Google Chrome, 2019)

Doplněk lze po publikaci najít na URL https://chrome.google.com/webstore/category/extensions.

#### 3.3.10.3 Ostatní prohlížeče

Posloupnost kroků pro publikaci na storu Opery do repozitáře https://addons.opera.com/cs/extensions/ jsou vystaveny na webové adrese[47], kde je samotná publikace bezplatná. (Dev.Opera, 2019)

Pro Microsoft Edge je návod na stránce URL https://docs.microsoft.com/en-us/microsoft-edge/extensions/getting-started.

## 3.4 React

React je JavaScriptová knihovna pro vytváření uživatelského rozhraní.

Mezi hlavní přednosti Reactu je deklarativní zápis, komponentová tvorba a přenositelnost kódu (Facebook Inc., 2019).

[45] Dokumentace k publikaci doplňku pro Firefox https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/Distribution/Submitting_an_add-on
[46] Postup publikace do Chrome webstore https://developer.chrome.com/webstore/publish
[47] Návod k publikaci pro Opera prohlížeč https://dev.opera.com/extensions/publishing-guidelines/

### 3.4.1 Deklarativní zápis

Díky tomuto zápisu je kód lépe pochopitelný, předvídatelný a rovněž lépe laditelný. Pro ladění lze nainstalovat do prohlížeče rozšíření. (Facebook Inc., 2019)

### 3.4.2 Komponentová tvorba

Díky knihovně se vytvářejí komponenty, které se dají skládat. Komponenty mají svůj stav a vstupní parametry. (Facebook Inc., 2019)

### 3.4.3 Znovupoužitelnost

React nabízí spoustu možností, jak a kam hotové komponenty vykreslit. Běžně se používá pro webové aplikace a umožňuje jej knihovna React-DOM. Stejný kód lze aplikovat i pro mobilní aplikace (iOS, Android) v podobě React Native. Existuje sada dalších knihoven pro renderování do různých zařízení. (Facebook Inc., 2019)

```
ReactDOM.render(element, document.getElementById('root'));
```

### 3.4.4 Charakteristika

React umožňuje vývojářům odstínit práci s DOMem, na rozdíl od jQuery. Každá komponenta je JavaScriptová třída se syntaxí ECMAScript 2017, která dědí třídu Component či další třídy.

Díky další knihovně JSX[48] lze obsah komponent psát velmi jednoduše díky tomu, že knihovna nabízí syntakticky jednoduší zápis komponent. Kód vypadá jako HMTL jen s tím rozdílem, že je to stále JavaScript a některé atributy se jmenují a zapisují jinak. (Facebook Inc., 2019)

Komponenty mohou mít svůj stav, ale nemusí. Dále mohou přijímat parametry zvenku a označují se jako „***props***“. Pro hlouběji zanořené komponenty je užitečné předávat kontext, se kterým lze pak pracovat jednodušeji než předávat ***props*** přes několik komponent.

Každá komponenta má svůj životní cyklus (*life-cycle*[49]), který je pro programátory vystaven pomocí React API.

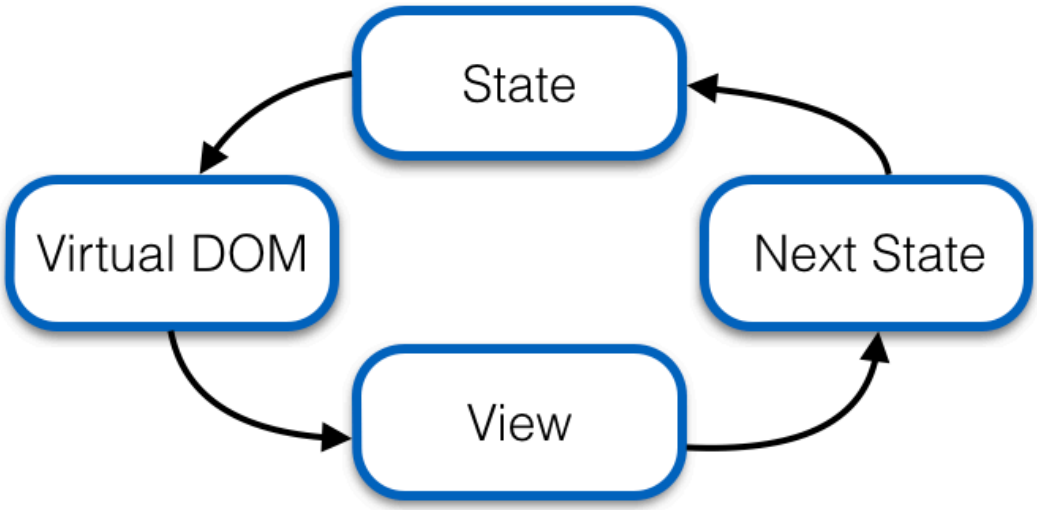
[48] https://reactjs.org/docs/introducing-jsx.html
[49] O life-cycle API na URL https://reactjs.org/docs/state-and-lifecycle.html

React funguje tak, že si nejdříve sestavuje svůj vlastní virtuální DOM (VDOM) a ten se renderuje do DOMu, až když je potřeba. Proto je rychlost vykreslování tak vysoká a zvyšuje se uživatelská přívětivost. Více o metodice překreslování na URL[50].

Jedinou povinně implementovanou metodou v rámci API je metoda ***render***. Ta slouží pro zápis těla komponenty, tedy co má komponenta vrátit za elementy, případně další komponenty. (Facebook Inc., 2019)

Komponenta vrací obsah metody ***render***, pokud se změní stav komponenty nebo vstupní parametr, tak se metoda zavolá znovu. S tím je spojeno i volání různých *life-cycle* metod, které lze implementovat v každé komponentě.

Každá komponenta je funkcí a tato skutečnost velmi nahrává paradigmatu funkcionálního programování. Koncept Reactu je inspirován tímto paradigmatem. Velkou roli hraje také neměnnost stavu (***immutabilita***) a to kvůli rychlosti přerenderování komponent. Na stejné úrovni je i vlastnost, jestli je komponenta tzv. Pure, což je jeden ze základních konceptů funkcionální programování. (Facebook Inc., 2019)




**Obrázek 16: Schéma fungování React knihovny (zdroj: Abernathy, 2015)**

[50] Metodika překreslování React komponent na URL https://reactjs.org/docs/reconciliation.html

#### 3.4.4.1 Stav

Stav reprezentuje objekt ***state***, který je definován programátorem. Tento objekt lze měnit pomocí metod v různých *life-cycle* metodách a iniciovat jej lze v konstruktoru.

Jakmile se stav změní, dojde k zavolání metody ***render*** v komponentě, a to vede k překreslení komponenty. (Facebook Inc., 2019)

#### 3.4.4.2 Props

Parametry komponenty jsou označovány jako ***props*** a vstupují zvenčí. Když dojde ke změně některého z nich, má to stejný důsledek jako změna stavu. Tyto parametry jsou pouze pro čtení.

***Props*** je také objekt, který může obsahovat několik atributů a je čistě na programátorovi, jak je pojmenuje. Nicméně existuje jedna rezervovaná ***props*** a ta se nazývá ***children***. Jde o obsah, který je zapsán jako tělo elementu (komponenty). (Facebook Inc., 2019)

Definice těchto parametrů lze zapsat a vynutit pomocí knihovny ***prop-types***[51] či statického typové kontroly ***FlowType***[52] nebo dokonce ***TypeScriptu***[53].

#### 3.4.4.3 Druhy komponent

Dle existence stavu v komponentách existuje několik druhů komponent.

##### Komponenta

Tento typ komponenty disponuje vlastním stavem, který si lze definovat, případně přijímá ***props*** a lze implementovat všechny *life-cycle* metody. (Facebook Inc., 2019)

```
class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

[51] Knihovna Proptypes https://reactjs.org/docs/typechecking-with-proptypes.html#proptypes
[52] Knihovna Flow https://flow.org/
[53] Knihovna Typescript https://www.typescriptlang.org/

### Pure komponenta

*Pure* komponenta může mít stav, přijímá ***props***. Chová se stejně jako obyčejná komponenta s tím rozdílem, že má optimalizovanou *life-cycle* metodu ***shouldComponentUpdate***, která se volá před metodou ***render*** a určuje, jestli se má metoda ***render*** zavolat. Obecně se doporučuje používat tento druh komponenty. (Facebook Inc., 2019)

```
class Welcome extends React.PureComponent {
  render() {
    return <h1>Welcome</h1>
  }
}
```

### Funkcionální (stateless) komponenta

Funkcionální komponenty lze označovat několika názvy (*stateless, funkcionální*), jejím hlavním rysem ale je, že nemá svůj stav. Má jen vstupní ***props***. Komponenta je zároveň *Pure*. (Facebook Inc., 2019)

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

Volání komponenty

```
<Welcome name="Sara" />
```

#### *3.4.4.4 Další vlastnosti*

React komponenty umožňují velmi jednoduše reagovat a odchytávat události, a to prostřednictvím *JSX* zápisu. Tímto odpadá přidávání listenerů do DOMu manuálně, protože React toto dělá automaticky.

```
function ActionLink() {
  function handleClick(e) {
    e.preventDefault();
    console.log('The link was clicked.');
  }

  return (
    <a href="#" onClick={handleClick}>
      Click me
    </a>
  );
}
```

V rámci komponenty může být i logika vykreslování, tedy že má funkce vykreslit nějaký výstup na základě nějaké podmínky stavu či ***props***. (Facebook Inc., 2019)

```
function Greeting(props) {
  const isLoggedIn = props.isLoggedIn;
  if (isLoggedIn) {
    return <UserGreeting />;
  }
  return <GuestGreeting />;
}

ReactDOM.render(
  // Try changing to isLoggedIn={true}:
  <Greeting isLoggedIn={false} />,
  document.getElementById('root')
);
```

Pro znuvupoužitelnost logiky uvnitř komponent lze využít vlastních komponent, které se nazývají *Higher-Order components*[54] případně nového API *React Hooks*[55]. (Facebook Inc., 2019)

React komponenta musí vždy vracet jednu hodnotu; pokud je potřeba vrátit více komponent, lze je obalit speciální komponentou Fragment[56], která nic nerenderuje do DOMu. (Facebook Inc., 2019)

Při dodržování psaní kódu dle výše zmíněného paradigmatu je potřeba přemýšlet jinak. Je dobré rozdělovat UI do hierarchie komponent, které jsou znovupoužitelné, ale dělají jen jednu věc a to správně. Také je záhodno držet stav aplikace jen na místech, kde je to výhodné a správné. K tomu lze využít ***state***, který nabízí sám React, nebo lze využít architektonické knihovny typu ***MobX***[57], ***Redux*** či ***Flux***[58]. ***Reduxu*** je věnována samostatná kapitola.

Více informací o knihovně React lze najít na webu dokumentace [59] i s příklady[60].

### 3.4.5 Redux

*Redux* je stavový kontejner pro JavaScriptové aplikace, který umožňuje předvídatelně definovat chování.

[54] O Higher-Order komponentách https://reactjs.org/docs/higher-order-components.html
[55] Detaily k React Hooks API https://reactjs.org/docs/hooks-intro.html
[56] Fragemnty v Reactu https://reactjs.org/docs/fragments.html
[57] Knihovna MobX https://mobx.js.org/
[58] Knihovna Flux https://facebook.github.io/flux/
[59] Dokumentace Reactu https://reactjs.org/docs/getting-started.html
[60] Návody a příklady k Reactu https://reactjs.org/tutorial/tutorial.html

Velmi často se používá v kombinaci s Reactem, který tvoří *View* vrstvu aplikace. (Abramov, 2019)

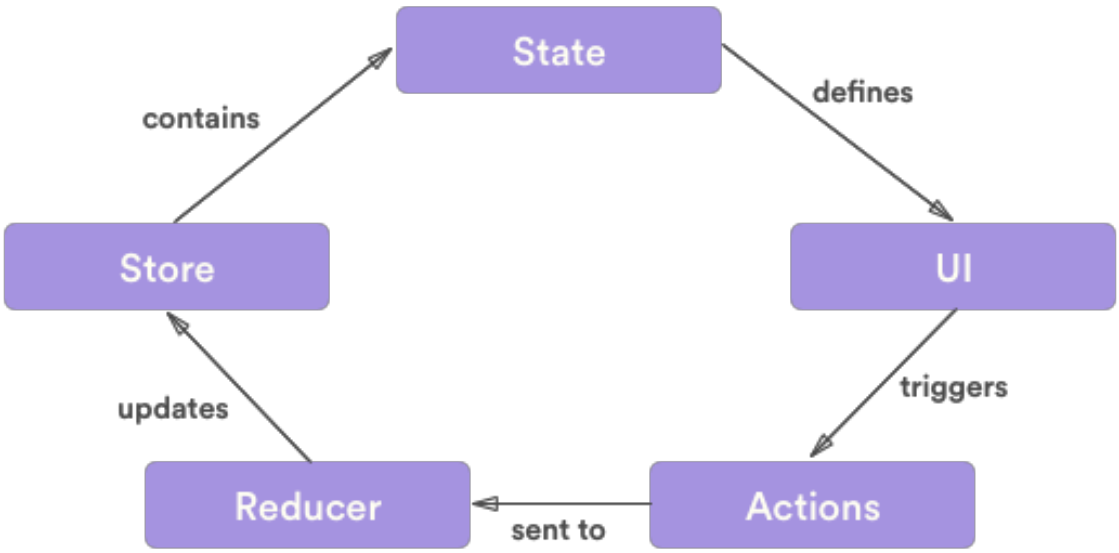
#### 3.4.5.1 Koncept a principy

Hlavními principy knihovny je udržovat stav jen na jednom místě čili mít jeden zdroj pravdy. Stav je tvořen objekty, které se jmenují ***state***. Tyto objekty lze sdružovat do stromu, který se nazývá ***Store***.

Tento stav aplikace je určený jen pro čtení, čímž se zajišťuje ***immutabilita*** (neměnnost) tohoto stavu, a to vede k lepší performance celé aplikace, protože React tuto vlastnost požaduje pro optimalizování komponent. Stav se tedy předává jako ***props*** do jednotlivých komponent a React ***state*** se většinou napoužívá. (Abramov, 2019)

Stav se dá ale měnit prostřednictvím volání akcí, které vrací objekty tvořené unikátním hodnotou atributu ***TYPE*** a samotných dat pro uložení do stavu. Akce lze vyvolat uvnitř i vně komponent. Dovnitř se předávají opět pomocí ***props***.

Na vyvolané akce pure funkce reagují jiné funkce, které stav mění a nazývají se reducer. Reducer naslouchá jako ***subscriber*** (Observer)[61] na akce a na základě konkrétní akce vrátí nový stav. Pokud se stav změní, dostane se jako ***props*** do *View* vrstvy (React komponent) a dojde k jejich překreslení. (Abramov, 2019)




**Obrázek 17: Popis chování knihovny Redux (zdroj: Tahir, 2018)**

[61] O vzoru Observer na URL https://en.wikipedia.org/wiki/Observer_pattern

*Redux* umožňuje přidat další knihovny díky respektování rozhraní ***middlewarů***. Ty se častou používají pro rozšíření funkcionality Reduxu. Ať už jde o lepší práci s asynchronními voláními (*Redux-thunk, Redux-saga*), práci s daty jako jsou *streamy (Redux-observables)* nebo pro debugování knihovny (*Redux-devtools*). Pro ladění lze nainstalovat do prohlížeče rozšíření Redux-devtools. (Abramov, 2019)

Pro jednotkové testování lze využít některý z nástrojů pro spouštění a psaní unit testů (např.: Jest).

#### *3.4.5.2 Kontejner komponenty*

Aby React komponenty mohly fungovat s Reduxem, je potřeba je s ním propojit. K tomu slouží funkce ***connect***, která přijímá jako parametry funkce pro namapování stavu na ***props*** a také akcí na ***props***. Posledním parametrem je pak samotná komponenta. (Abramov, 2019)

### *3.4.6 Bootstrap*

*Bootstrap* je nejpopulárnější front-end open-source knihovna pro tvorbu responzivních webových aplikací. Obsahuje CSS, HTML i JavaScript. Vývojáři tak mohou velmi rychle vyvíjet šablony díky použití již hotových komponent. (Bootstrap team, 2019)

CSS kód je v nejnovější verzi 4.x napsaný v preprocesoru SASS. Web[62] obsahuje velmi podrobnou a přehlednou dokumentaci.

V praktické části je použita právě nejnovější verze.

### *3.4.7 Reactstrap*

*Reactstrap* je open-source knihovna, která obsahuje React komponenty vytvořené podle Bootstrap knihovny. Kód je kompatibilní s novějšími verzemi Boostrapu, který se musí pro použití nainstalovat také. (Reactstrap, 2018)

Komponenty v této React knihovně jsou otestované jednotkovými testy a často se používají i v produkci různých projektů na internetu.

[62] Webová stránka projektu https://getbootstrap.com/

Rovněž projekt obsahuje i velmi povedenou dokumentaci [63] s příklady a vyhledáváním komponent, včetně přehledu ***props***, které komponenty mohou přijmout.

### 3.4.8 Abstraktní syntaktický strom (AST)

*AST* představuje abstraktní syntaktickou strukturu zdrojového kódu programovacího jazyka. Strom tvoří operátory jako uzly a operandy jako listy. Tento strom se v informatice používá pro překlad nebo optimalizaci kódu.

Tato datová struktura se také hojně používá u kompilátorů jazyků, protože udržuje strukturu a vztahy mezi instrukcemi. Jazyky často bývají specifikovány jako bezkontextové gramatiky

*AST* vzniká prostřednictvím syntaktické analýzy.

V případě vytvářené aplikace se bude jednat o *AST* jazyka CSS a to proto, aby bylo možno pracovat s jednotlivými selektory v kódu. (Wikipedia, 2018)

#### 3.4.8.1 CSS parser

Open-source knihovna [64] *CSS parser* umožňuje přijmout řetězec jako vstup (CSS kód) a vrací AST objekt. Jde tedy o parsování a jedná se o metodu ***parse***.

[63] Dokumentace knihovny na https://reactstrap.github.io/
[64] Odkaz na knihovnu CSS https://github.com/reworkcss/css

# 4 Praktická část

Vzhledem k tomu, že cílem práce je vytvořit nástroj pro vývojáře, zužuje se cílová skupina uživatelé na velmi specifickou množinu uživatelů.

Jelikož se jedná o open-source projekt, nepočítá se se žádnou monetizací, a tudíž je bezpředmětné vytvářet studii příležitosti.

Studie použitelnosti na místě je.

Výsledná aplikace bude lokalizována do anglického jazyka.

## 4.1 Akceptační kritéria

Pro určení toho, že je projekt dokončen je nezbytné splnit akceptační kritéria, která jsou definována takto:

- Aplikace validuje navštívenou webovou stránku uživatele dle metodiky BEM v rozhraní prohlížeče.
- Aplikace zobrazuje chybné selektory a chybějící CSS třídy s možností jednotlivé selektory zobrazit v průzkumníku prohlížeče.
- V případě chyby nebo detekce nepřítomnosti BEMu na stránce aplikace tuto informaci zobrazí.
- Při úspěšné validaci stránky zobrazí aplikace tuto skutečnost uživateli.

## 4.2 Persony

Definice tří person pro cílovou skupinu.

### *Pavel Neználek*

- Rodinný stav: zaměstnaný, ženatý
- Věk: 32 let
- Vzdělaní: SOŠ
- Bydliště: Brno
- Koníčky: přátelé, rodina, sport
- Národnost: česká
- Zaměstnání: full-stack programátor webů

- Zkušenost v oboru: Programoval na střední škole v C++, poté ale přesedlal na web. Už 10 let vytváří weby v agentuře. Metodiku BEM zná jen málo, ale rád by ji začal používat správně.
- Znalost jazyku: čeština, angličtina (B2), slovenština

### *Jakub Zkušený*

- Rodinný stav: svobodný
- Věk: 28 let
- Vzdělaní: vysoká škola
- Bydliště: Praha
- Koníčky: přátelé, publikace, konference, vzdělávání
- Národnost: česká
- Zaměstnání: programátor webového front-endu
- Zkušenost v oboru: Téměř 6 let vytváří weby v projektovém týmu jedné větší firmy. Metodiku BEM velmi dobře a zná její benefity při práci v týmu a udržování UI knihoven.
- Znalost jazyku: čeština, angličtina (C1), slovenština, němčina (B2)

### *Paul Newbie*

- Rodinný stav: svobodný
- Věk: 20 let
- Vzdělaní: střední škola
- Bydliště: New York, USA
- Koníčky: sport, PC, hudba
- Národnost: americká
- Zaměstnání: stážista ve IT firmě
- Zkušenost v oboru: Základy programování získal na univerzitě, kterou stále studuje kombinovaně, a souběžně pracuje jako stážista ve firmě, kde získává první zkušenosti s tvorbou webů. Týmy ve firmě BEM metodiku používají, ale Paul ještě ne.
- Znalost jazyku angličtina (rodilý mluvčí)

## 4.3 Use case

Přehled use casů pro aplikaci.

### *4.3.1 Validace webové stránky, která je již načtená a obsahuje chyby*

1) Uživatel otevře okno pro vývojáře a klikne na panel s doplňkem.
2) Systém zobrazí obrazovku s načítací animací.
3) Systém nalezne chyby a vypíše je v seznamu na obrazovku.

### *4.3.2 Validace webové stránky, která je již načtená a neobsahuje chyby*

1) Uživatel otevře okno pro vývojáře a klikne na panel s doplňkem.
2) Systém zobrazí obrazovku s načítací animací.
3) Systém vypíše na obrazovku informaci o validitě stránky.

### *4.3.3 Zobrazení konkrétního selektoru, který je zjištěn jako chybný při validaci webové stránky, která je již načtená*

1) Uživatel otevře okno pro vývojáře a klikne na panel s doplňkem.
2) Systém zobrazí obrazovku s načítací animací.
3) Systém nalezne chyby a vypíše je v seznamu na obrazovku.
4) Uživatel klikne na odkaz pro selektor v seznamu.
5) Systém zobrazí panel s elementy prohlížeče na konkrétním elementu.

### *4.3.4 Validace webové stránky, na kterou se přechází s otevřeným doplňkem po existující validaci*

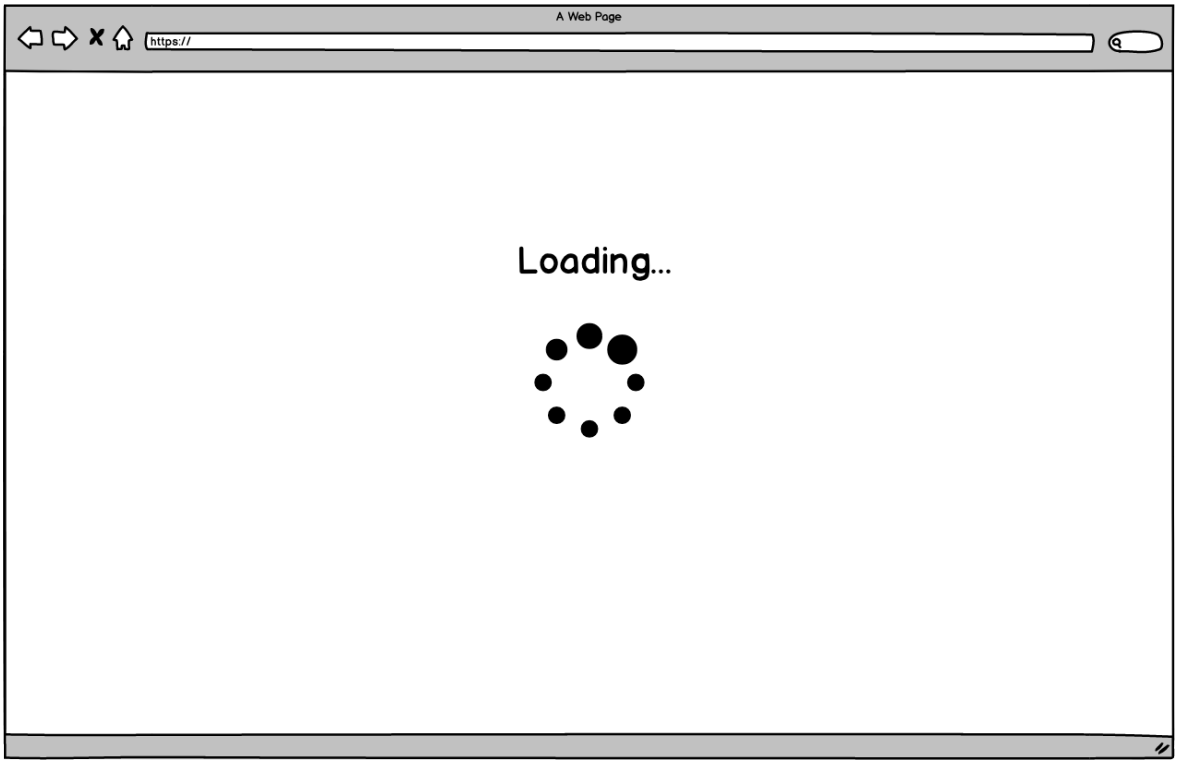
1) Uživatel přejde na jinou webovou stránku (odkazem nebo změnou URL).
2) Systém vymaže předchozí data z obrazovky.
3) Systém zobrazí obrazovku s načítací animací.
4) Systém vypíše na obrazovku informaci o validitě stránky.

### *4.3.5 Validace webové stránky, která je obnovována s otevřeným doplňkem po již existující validaci*

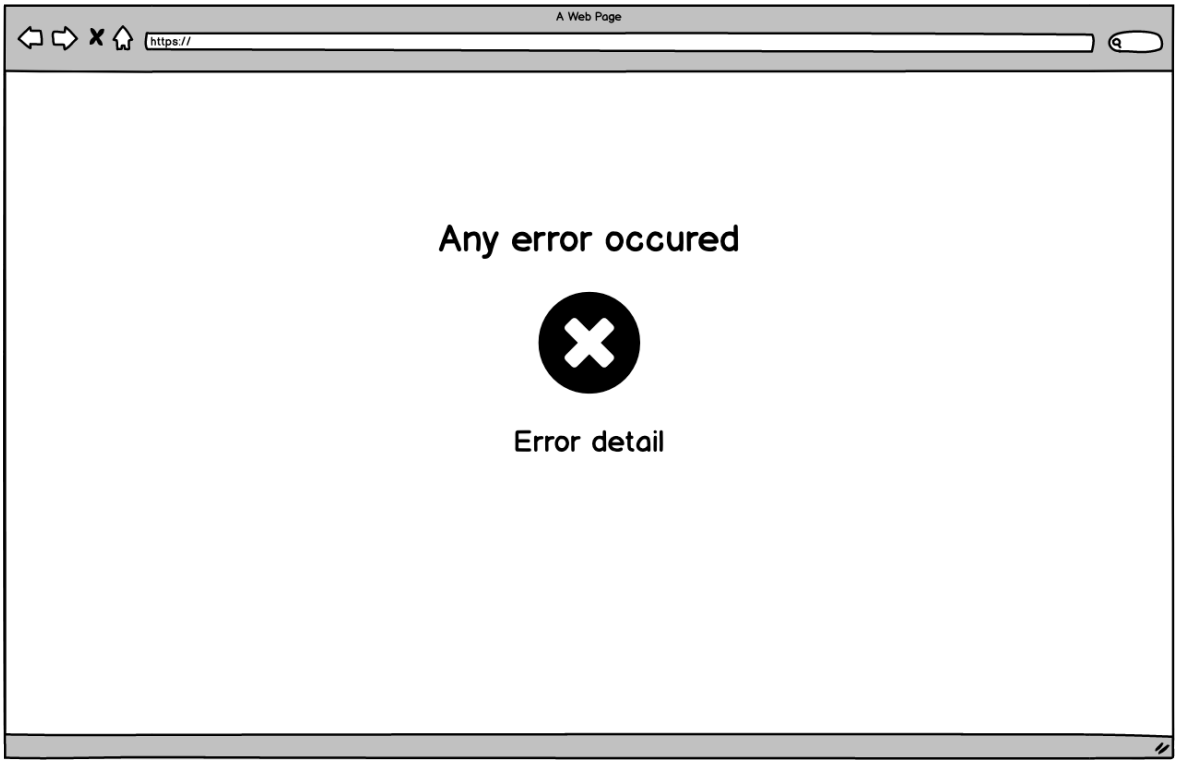
1) Uživatel obnoví stránku (F5).
2) Systém vymaže předchozí data z obrazovky.
3) Systém zobrazí obrazovku s načítací animací.
4) Systém vypíše na obrazovku informaci o validitě stránky.

## 4.4 Wireframy

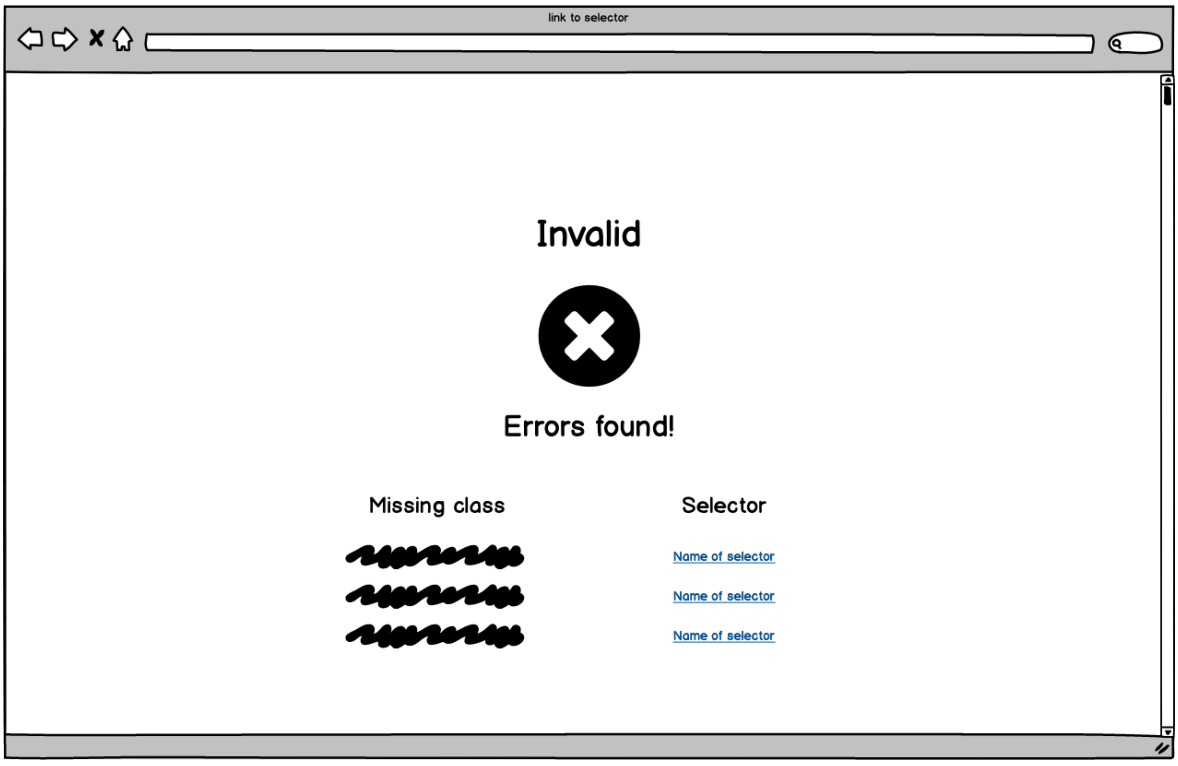
Přehled wireframů pro jednotlivé stavy aplikace.



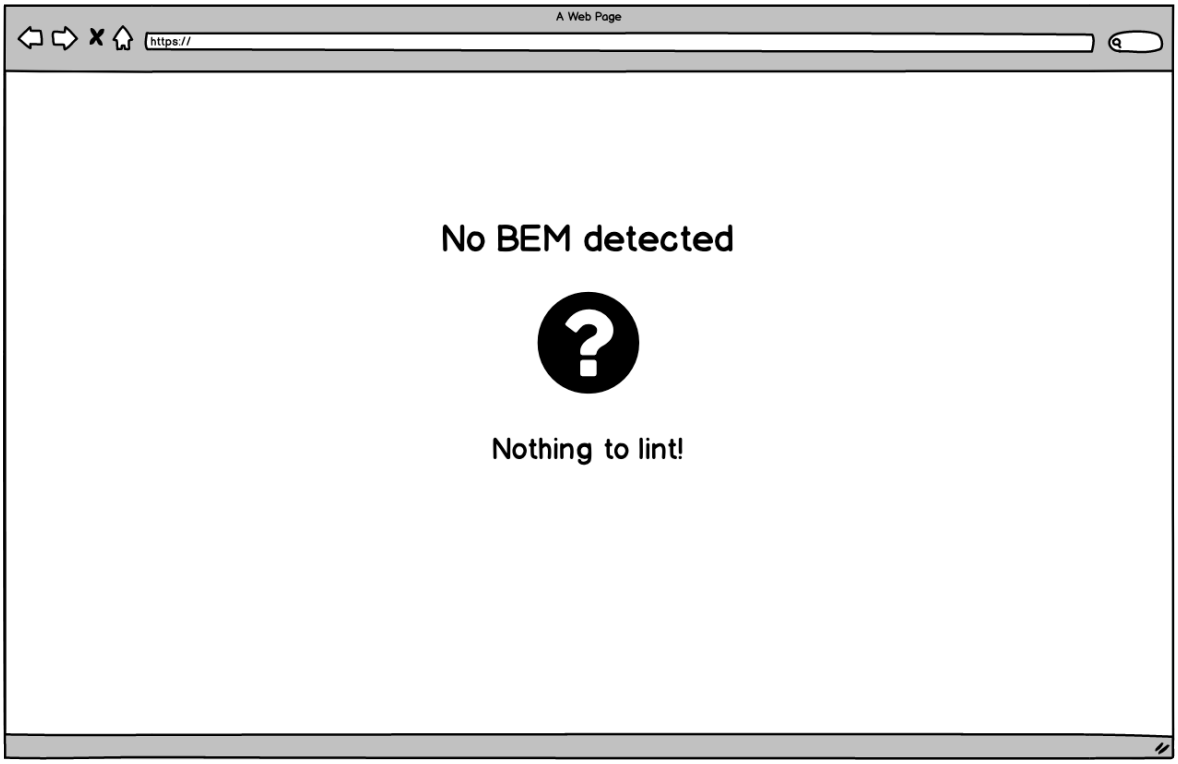

**Obrázek 18: Stav načítání (zdroj: vlastní)**



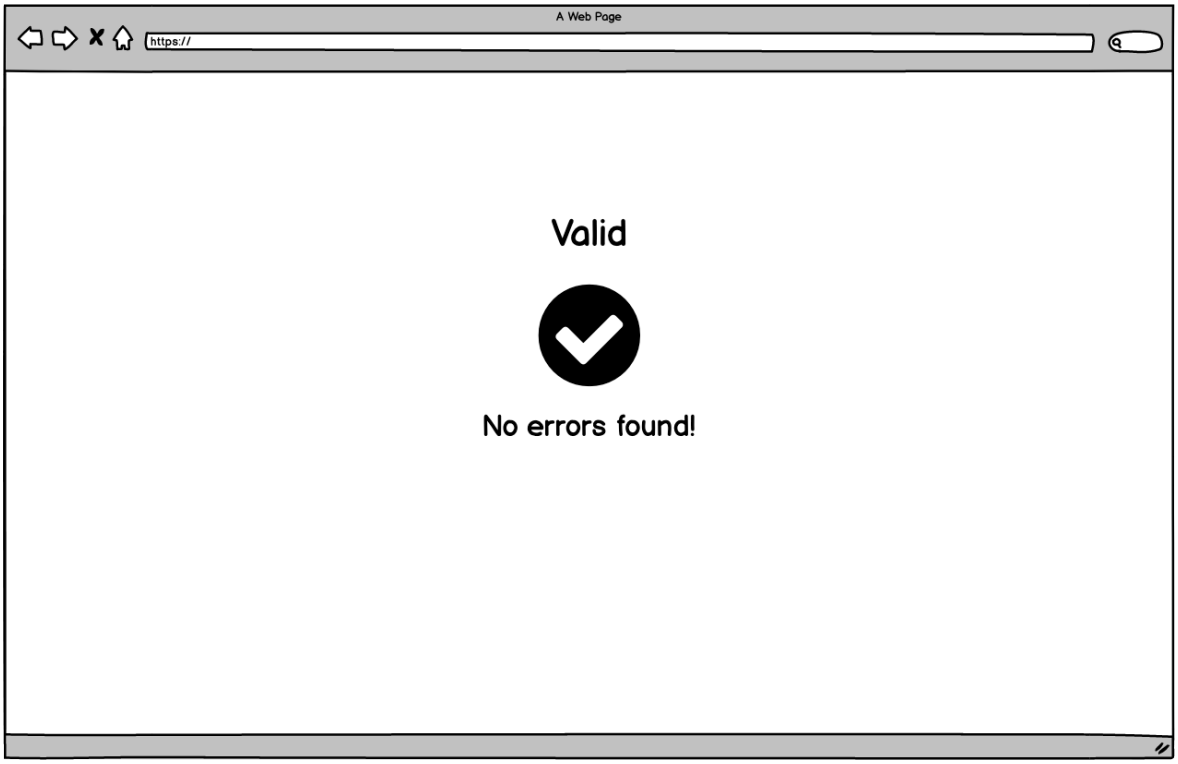

**Obrázek 19: Stav v případě neočekávané chyby (zdroj: vlastní)**




**Obrázek 20: Nevalidní stav (zdroj: vlastní)**




**Obrázek 21: Stav v případě, že není co validovat (zdroj: vlastní)**




**Obrázek 22: Validní stav (zdroj: vlastní)**

## 4.5 Návrh řešení

Před samotnou implementací je nutno navrhnou řešení, a to do následujících kroků:

- Rozšíření do prohlížeče získá z DOMu navštěvované stránky všechny použité CSS třídy.
- Tyto třídy zparsuje do abstraktního syntaktického stromu (AST).
- Tento strom předá jako vstup do lintovací knihovny *CSS-should-plugin-bem*, která provede validaci oproti metodice BEM. Knihovna se bude muset upravit oproti původní verzi, kdy nebyla schopna validovat jinak než ze souboru.
- Knihovna vrátí seznam selektorů, u kterých byla nalezena chyba a s ním i CSS třídy, které je potřeba doplnit pro validní BEM zápis.
- Informace vypíše uživateli do okna doplňku.
- V případě nalezených chyb nabídne možnost kliknutí na detail selektoru v nativním panelu prohlížeče.

## 4.6 Architektura aplikace

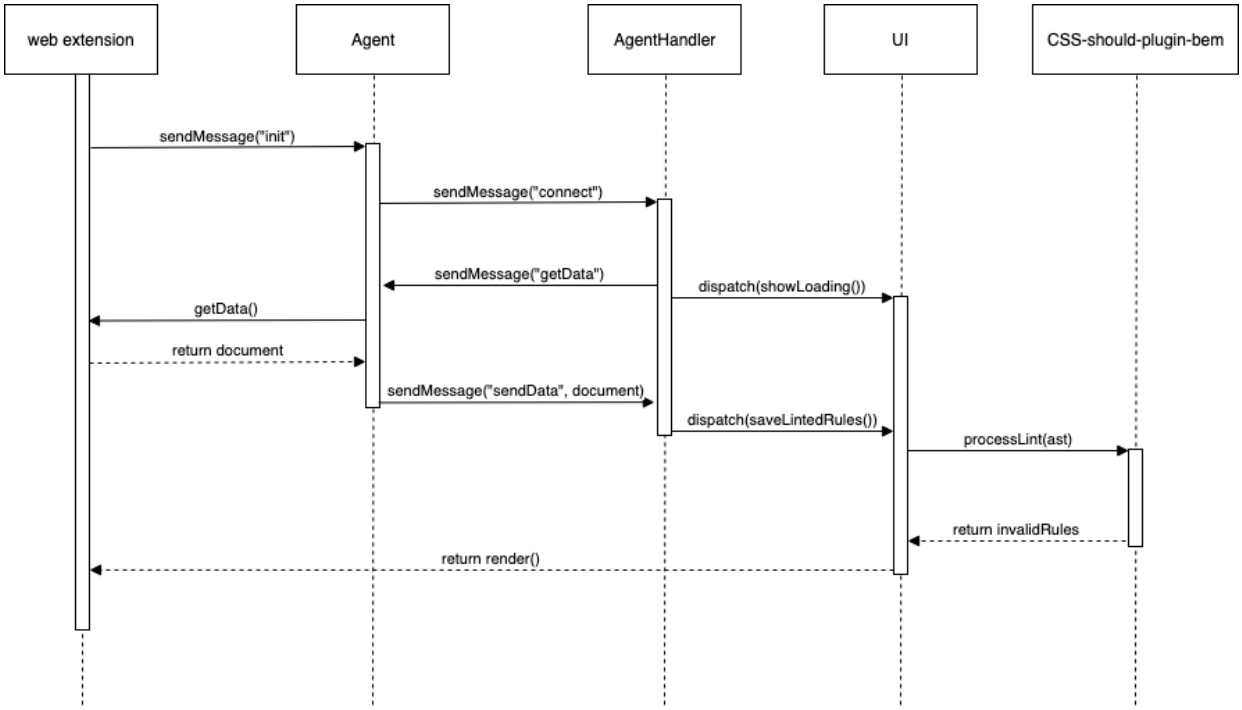
Aplikace běží ve webovém prohlížeči a komunikuje synchronně s externí knihovnou *CSS-should-plugin-bem*, která je pro klíčovou funkčnost nezbytná. Dále využívá spoustu závislostí, které jsou použity pro vytvoření aplikace.

Aplikace obsahuje několik vrstev:

- Vrstva pro komunikaci s prohlížečem (*web extension*).
- Vrstva pro komunikaci s web extension (*Agent*).
- Vrstva aplikace, která je pomocí webextension zobrazena (*UI*).
- Vrstva třetí strany – knihovna *CSS-should-plugin-bem*.
- Vrstva pro komunikaci mezi Agentem a UI (*AgentHandler*).

Pro samotný vývoj bylo použito jazyku JavaScript a jeho serverové verze Node.js (verze v9.11.2). K instalaci balíčku byl využit balíčkovací manager NPM ve verzi 6.5.0.

### 4.6.1 Sekvenční diagram

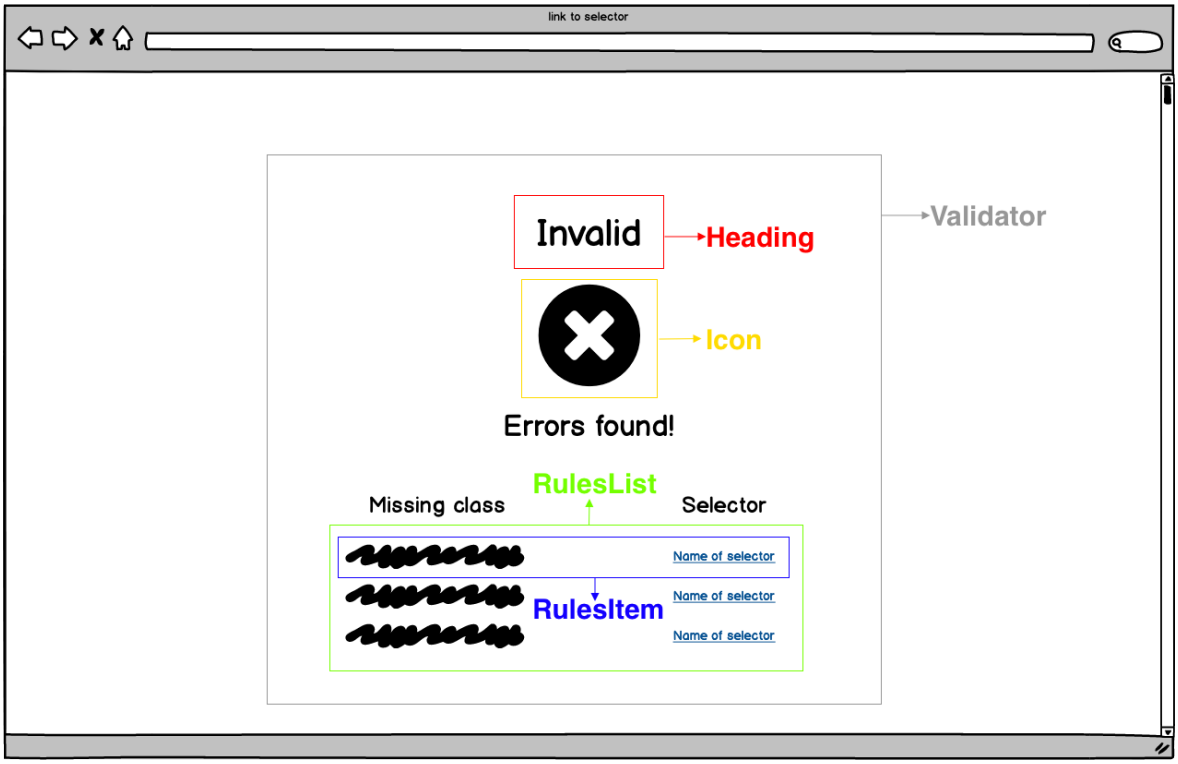



**Obrázek 23: Sekvenční diagram aplikace při nalezení nevalidních dat (zdroj: vlastní)**

## 4.7 Návrh tříd

Jelikož použité technologie spíše podporují funkcionální paradigma, tak klasických tříd OOP v projektu moc není. Spousta funkcionality jsou jen moduly, které obsahují funkce, které jsou exportované a importují se na místech volání.

View vrstvu lze rozpadnout do React komponent (tříd) dle obrázku.




**Obrázek 24: Návrh komponent na základě wireframu (zdroj: vlastní)**

Vznikne několik komponent, které lze znovu využít.

### 4.7.1 React komponenty

Rozdělení komponent pro React na základě wireframů a jejich popis.

#### 4.7.1.1 Heading

Komponenta ***Heading*** je určena pro nadpis, který se používá pro všechny stavy aplikace. Přijímá jako parametr obsah typu řetězec.

##### 4.7.1.1.1 Icon

***Icon*** komponenta je komponentou s *SVG* zdroji, která se renderuje pro všechny stavy s odlišným zdrojovým obrázkem (ikonou) dle parametru. Přijímá povinný parametr „***kind***“ pro rozlišení zdroje.

#### 4.7.1.2 RulesItem

Řádek tabulky se svými specifiky pro použití je definován v komponentě ***RulesItem***. Může se vyskytovat na stránce několikrát. Vstupními parametry je samotný objekt pravidla (***missingClassName*** a ***selector***) či funkce, která se vykoná při kliknutí na text selektoru.

```
RulesItem.propTypes = {
  rule: PropTypes.exact({
    missingClassName: PropTypes.string.isRequired,
    selector: PropTypes.string.isRequired,
  }).isRequired,
  inspectElement: PropTypes.func.isRequired,
};
```

#### 4.7.1.3 RulesList

***RulesList*** komponenta reprezentuje kolekci ***RulesItem*** komponent. Vstupuje do ní kolekce pravidel „***rules***“ a funkce, která se předává potomkům.

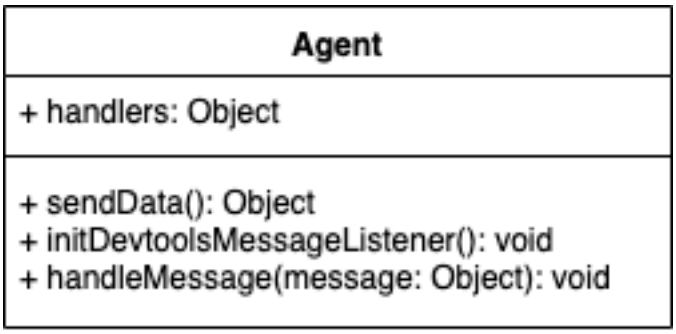
```
RulesList.propTypes = {
  rules: PropTypes.arrayOf(
    PropTypes.exact({ missingClassName: PropTypes.string.isRequired,
selector: PropTypes.string.isRequired }),
  ).isRequired,
  inspectElement: PropTypes.func.isRequired,
};
```

#### 4.7.1.4 Validator

Komponenta ***Validator*** je stěžejní pro renderování obsahu dle stavu aplikace. Zde se rozhoduje, které komponenty, a s jakým obsahem, budou zavolány.

```
Validator.propTypes = {
  rules: PropTypes.arrayOf(
    PropTypes.exact({ missingClassName: PropTypes.string.isRequired,
selector: PropTypes.string.isRequired }),
  ).isRequired,
  loading: PropTypes.bool.isRequired,
  error: PropTypes.string.isRequired,
  isBemDetected: PropTypes.bool.isRequired,
  isValid: PropTypes.bool.isRequired,
  inspectElement: PropTypes.func.isRequired,
};
```

### 4.7.2 Agent

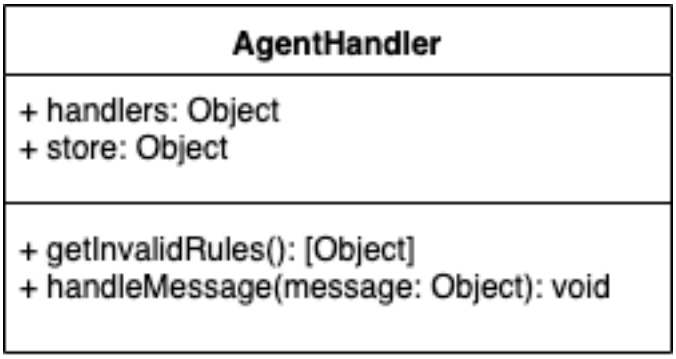



**Obrázek 25: Class diagram třídy Agent (zdroj: vlastní)**

Třída ***Agent*** reprezentuje vrstvu pro komunikaci mezi web extension a stránkou. Naslouchá od počátku své existence na zprávy a operuje s DOMem navštívené stránky.

### 4.7.3 AgentHandler

Třída ***AgentHandler*** je určena komunikace s ***Agentem*** pomocí zpráv, disponuje funkcionalitou pro manipulace se stavem aplikace (store) prostřednictvím volání Redux akcí. Také slouží pro komunikaci s knihovnami třetích stran, tedy hlavně s knihovnou pro lintování (*CSS-should-plugin-bem*).




**Obrázek 26: Class diagram třídy AgentHandler (zdroj: vlastní)**

### 4.7.4 Redux

Pro uchovávání stavu aplikace a jeho změny je záhodno pro Redux vytvořit počáteční stav a také akce, které jej budou měnit.

```
{
  rules: [],
  loading: true,
  error: '',
  isValid: false,
  isBemDetected: false,
};
```

Mimo toto je potřeba nastavit i akce, které budou stav měnit prostřednictvím ***reduceru***.

### 4.7.5 Web extension

Pro vytvoření doplňku, kam pak bude výše zmiňovaná část aplikace vykreslena je potřeba také navrhnout architekturu.

K vykreslování panelu do *devtools* poslouží vytvoření API ***devtools.panels.create()*** v adekvátním souboru.

Komunikace s obsahem bude zajištěna pomocí *obsahového skriptu (content script)*. Obdobně tomu bude v případě výměny zpráv mezi *devtools* a *obsahovým skriptem*, a to skrz *background skript (skript na pozadí)*, který bude mít na starosti i stav otevřeného panelu prohlížeče.

Tyto dva soubory jsou stěžejní pro posílání zpráv mezi jednotlivými vrstvami.

### 4.7.6 CSS-should-plugin-bem

Knihovna *CSS-should-plugin-bem* musí být upravena tak, aby byla schopna prostřednictvím svého API přijmout AST kaskádových stylů a lintovat jej.

Hlavní funkcionalitou bude vylepšení funkce ***lint***, která nebude vracet jen seznam chybějících tříd jako doposud, ale rovnou kolekci selektorů a chybějících tříd.

Dále je nezbytné vrátit informace o tom, že validace vůbec proběhla a jestli tedy úspěšně nebo ne, jelikož AST nemusí obsahovat žádné BEM třídy.

## 4.8 Implementace

V této části závěrečné práce přichází na řadu samotné vytváření kódu, případně upravování existující aplikace *CSS-should-plugin-bem*.

Prvním krokem bylo vytvořit adresářovou strukturu projektu, nastavování prostředí, konfiguračních souborů a instalace pomocných knihoven pro vývoj.

### 4.8.1 Struktura projektu

#### 4.8.1.1 Kořenový adresář

Stěžejním souborem projektu v kořenovém adresáři je soubor *package.json*, který obsahuje základní metadata projektu nejen pro případnou publikaci balíčku v repozitáři NPM, ale také pro spuštění skriptů a seznam závislostí.

Nedílnou součástí projektu je také verzování samotného kódu. K tomu posloužil nástroj GIT, pro který existuje rovněž soubor pro ignorování souborů pro verzování *.gitignore*. Typicky se do souboru zahrnují adresáře s instalovanými závislostmi a také generovaný kód. Repozitář je umístěn na Githubu[65].

Pro formátování kódu byl použit nástroj Prettier a ke statické analýze byly použity nástroje ESLint a Stylelint. Všechny zmíněné nástroje potřebují k fungování svoje konfigurační soubory v projektu.

K tomu, aby bylo možno využívat moderní konstrukce ECMAScriptu či knihovny React, je nezbytný nástroj Babel, který transformuje novou podobu kódu do verze, které Node.js či prohlížeče rozumí a umí jej vykonat. Konfigurace se nachází v souboru *.babelrc*.

K vyvíjení aplikace v JavaScriptu je potřeba mít k dispozici vývojový server. K tomuto účelu se hodí balíček ***dev-server***, který je součástí nástroje webpack. Nástroj slouží k balení modulů, zdrojových souborů, obrázků či stylů do několika možných výstupních souborů dle konfigurace, kterou je možno si upravit dle potřeby v souboru *webpack.config.js*. Ke správnému balení těchto všech souborů musí být doinstalovány a dodefinovány odpovídající závislosti tzv. loadery či pluginy.

Nedílnou součástí projektu je i soubor *README.md*, který obsahuje základní informace o projektu s návody, jak s aplikací pracovat.

[65] Odkaz na repozitář na Githubu https://github.com/MartinKristof/bem-validator-devtools-extension

Funkci pro přehled změn v jednotlivých verzích aplikace slouží soubor *CHANGELOG.md*. V obou zmíněných souborech se pro zápis používá značkovací jazyk ***Markdown***.

#### 4.8.1.2 Složky

V kořenovém adresáři se nachází několik složek.

##### 4.8.1.2.1 Webextension

*Webextension* ***je*** nejdůležitější složkou, ve které jsou umístěny soubory pro funkci jako doplňku do prohlížeče. Kromě zmiňovaného souboru *manifest.json* obsahuje i soubory background.js, content-script.js, devtools.js, devtools.html, panel.html a ikonku (icon.png). Tyto soubory nejsou nikterak modifikovány *Babelem*, s nimi pracuje přímo prohlížeč.

##### 4.8.1.2.2 Src

Zdrojová složka *src* pro vrstvy Agent a UI, které jsou obdobně dle názvu rozděleny. Obsahují několik pomocných souborů s funkcemi pro posílání zpráv.

Složka agent dále obsahuje soubor Agent.js, který slouží ke komunikaci mezi webextension a aplikací ve složce *UI*.

Uvnitř adresáře UI se vyskytuje samotná React-Redux aplikace jako view vrsta UI aplikace. Dále jsou tu soubory pro vložení skriptu do navštívené stránky a také soubor *AgentHandler.js* pro komunikaci s Agentem. Nechybí ani inicializační skript UI aplikace. Pro separování CSS tříd z DOMu existuje soubor ***extractCss***.

##### 4.8.1.2.3 Styles

Složka se stylopisy pro SASS preprocesor.

##### 4.8.1.2.4 Tests

Ve složce *tests* jsou umístěny jednotkové testy pro aplikaci. Pro testování byl použit nástroj Jest. Nástroj umožňuje generovat i statistiku pokrytí jednotlivých souborů testy.

#### 4.8.1.2.5 Assets

Prostor pro zdrojové soubory typu obrázek apod., které jsou používání jako vstupní pro pozdější práci je ve složce *assets*. V případě projektu jsou to ikonky ve formátu SVG, ze kterých se pak generují React komponenty pomocí knihovny *react-svg-icon-generator*.

### 4.8.2 Aplikace React-Redux

Prvním krokem pro vyvinutí aplikace je vytvoření grafického rozhraní aplikace, která se až nakonec bude integrovat s web extension.

#### 4.8.2.1 React

##### 4.8.2.1.1 Komponenty

Pro rychlejší vývoj je výhodnější si vytvořit nejprve React aplikaci, která bude nezávislá na web extension a kterou není moc pracné otestovat pomocí jednotkových testů.

Do složky *ui* se vytvoří složka *components*, která bude obsahovat komponenty odpovídající návrhu.

Pro ukázku je nejzajímavější komponenta ***Validator***, která reprezentuje hlavní logiku pro vykreslování stavů.

```
const Validator = ({ rules, loading, error, isBemDetected, isValid,
inspectElement }) => (
  <Container className="text-center">
    <Row className="my-4">
      <Col>
        {!loading && error && (
          <Fragment>
            <Heading>Any error occured!</Heading>
            <Icon kind="error" />
            <p className="secondary">{error}</p>
          </Fragment>
        )}
        {loading && (
          <Fragment>
            <Heading>Loading...</Heading>
            <Icon kind="loader" />
          </Fragment>
        )}
        {!loading && rules.length > 0 && !error && isBemDetected &&
!isValid && (
          <div>
            <Heading>Invalid!</Heading>
            <div>
              <Icon kind="invalid" />
```

```
              <p className="error">BEM errors found!</p>
            </div>
            <Table responsive striped>
              <thead>
                <tr>
                  <td>Missing class</td>
                  <td>Selector</td>
                </tr>
              </thead>
              <tbody>
                <RulesList rules={rules} inspectElement={inspectElement}
/>
              </tbody>
            </Table>
          </div>
        )}
        {!loading && !error && isBemDetected && isValid && (
          <Fragment>
            <Heading>Valid!</Heading>
            <div>
              <Icon kind="valid" />
              <p className="success">No BEM errors found!</p>
            </div>
          </Fragment>
        )}
        {!loading && !error && !isBemDetected && !isValid && (
          <Fragment>
            <Heading>No BEM classes detected!</Heading>
            <Icon kind="unknown" />
            <p className="secondary">Nothing to lint!</p>
          </Fragment>
        )}
      </Col>
    </Row>
  </Container>
);

Validator.propTypes = {
  rules: PropTypes.arrayOf(
    PropTypes.exact({ missingClassName: PropTypes.string.isRequired,
selector: PropTypes.string.isRequired }),
  ).isRequired,
  loading: PropTypes.bool.isRequired,
  error: PropTypes.string.isRequired,
  isBemDetected: PropTypes.bool.isRequired,
  isValid: PropTypes.bool.isRequired,
  inspectElement: PropTypes.func.isRequired,
};
```

V kódu lze vidět v poslední části výčet ***props***, které komponenta vyžaduje, a zdali jsou povinné. Za zmínku stojí atribut ***rules***, který reprezentuje pravidla, která jsou vrácena lintovacím procesem a reprezentují nevalidní selektory a chybějící třídy v DOMu vůči BEMu.

Uvnitř těla komponenty jsou využívány komponenty knihovny Reactstrap pro využití Bootstrap UI v React aplikaci. Mimo ně jsou volány i vlastní komponenty (***Icon*** –

generovaná komponenta s SVG obrázkem, ***Heading*** – nadpis libovolné úrovně a ***RulesList*** – kolekce, která iterativně vypisuje ***RulesItem*** komponentu).

Poslední zmíněná komponenta slouží pro výpis tabulky s možností provedení akce při kliknutí na selektor, která je předávána jako ***props***.

React komponenty tedy v tomto případě stav vůbec neudržují a všechny informace přijímají jako vstupní ***props***.

#### *4.8.2.1.2 Jednotkové testy*

Takto vytvořenou aplikaci lze jednotkově otestovat pomocí Jest a Enzyme knihovny, a to efektivně pomocí snapshot testů pro každou komponentu.

Soubory se nachází ve složce *tests* a následně jsou zanořené tak, že kopírují umístění testovaných souborů.

```
  it('should render if rules are invalid according BEM', () => {
    const rules = [
      { missingClassName: '.foo', selector: '.bar' },
      { missingClassName: '.boo', selector: '.bazz' }
    ];

    const node = shallow(
      <Validator
        rules={rules}
        isBemDetected
        isValid={false}
        loading={false}
        inspectElement={inspectElement}
        error=""
      />,
    );

    expect(node).toMatchSnapshot();
  });
```

### *4.8.2.2 Redux*

K udržování stavu aplikace a jeho modifikací je záhodno vytvořit počáteční stav a ten napojit na React aplikaci.

Počáteční stav je definován v návrhu, ale je potřeba připojit jej do ***store***.

#### *4.8.2.2.1 Akce*

Pro změnu stavu se musí definovat akce typicky do složky *actions.js*. Akce přijímají parametry a vrací je společně s unikátním typem.

```
export const saveLintedRules = ({ rules, isBemDetected, isValid }) => ({
  type: ACTION_SAVE_LINTED_RULES,
  rules,
  isBemDetected,
  isValid,
});

export const showLoading = () => ({
  type: ACTION_SHOW_LOADING,
});

export const showError = (error) => ({
  type: ACTION_SHOW_ERROR,
  error,
});

export const inspectElement = (element) => {
  inspect(element);

  return {
    type: ACTION_INSPECT_ELEMENT,
    element,
  };
};
```

Speciální akcí je ***inspectElement***, která jako side-effect volá funkci pro ovládání prohlížeče. Konkrétně jde o vykonání nativní funkce ***inspect*** pro selektor v DOMu.

```
export const inspect = (selector) => {
  if (!selector) {
    return;
  }

  const inspectString = `inspect(document.querySelector('${selector}'))`;

  try {
    browser.devtools.inspectedWindow.eval(inspectString);
  } catch (error) {
    const sendMessage = require('./util/sendMessage');

    sendMessage('error', error.message);
  }
};
```

#### 4.8.2.2.2 Reducer

Výše zmíněné akce jsou pomocí unikátních typů odchytávány v ***reduceru***, který se pak stará o změnu stavu. Nejzajímavější změnou stavu je řízení akce *ACTION_SAVE_LINTED_RULES*.

```
const initialState = stateRecord;

export default (state = initialState, action) => {
  switch (action.type) {
    case ACTION_SAVE_LINTED_RULES:
      return {
        ...state,
        rules: action.rules || [],
        loading: false,
        isValid: action.isValid || false,
        isBemDetected: action.isBemDetected || false,
      };
    case ACTION_SHOW_ERROR:
      return { ...state, error: action.error };
    case ACTION_SHOW_LOADING:
      return { ...state, loading: true };
    default:
      return state;
  }
};
```

#### 4.8.2.2.3 Kontejner

Redux aplikace je sice vytvořena, ale ještě není napojena na React aplikaci a k tomu poslouží kontejner komponenta *index.js*.

```
import { connect } from 'react-redux';
import { bindActionCreators } from 'redux';

import Validator from './Validator';
import { inspectElement } from '../actions';

const mapStateToProps = ({ state }) => ({
  ...state,
});

const mapDispatchToProps = (dispatch) => ({
  inspectElement: bindActionCreators(inspectElement, dispatch),
});

export default connect(
  mapStateToProps,
  mapDispatchToProps,
)(Validator);
```

Zde jsou dvě klíčové metody pro definici, pomocí kterých se mapuje stav na ***props***, a toto samé se děje s akcemi.

Komponenta vrací napojenou komponentu ***Validator*** na stav, a to pomocí funkce ***connect()*** z balíčku react-redux.

#### *4.8.2.2.4 Jednotkové testy*

Jednotkové testy pro Redux vrstvu (*actions.js, reducer.js, state.js a index.js*) lze rovněž otestovat stejným způsobem, zde ale není potřeba dělat snapshot testing (kromě kontejner komponenty), protože se operuje s funkcemi, které jen vrací data a jsou idempotentní.

```
describe('#saveLintedRules', () => {
    it('should call action with params', () => {
      const expectedAction = {
        type: ACTION_SAVE_LINTED_RULES,
        rules: [{ foo: '.bar' }],
        isBemDetected: true,
        isValid: false,
      };

      const payload = { rules: [{ foo: '.bar' }], isBemDetected: true,
isValid: false };
      expect(saveLintedRules(payload)).toEqual(expectedAction);
    });
  });
```

#### *4.8.2.2.5 Připojení do DOMu*

Následujícím způsobem se vytvoří a připojí React-Redux aplikace do DOMu na konkrétní element včetně připojení vlastních stylů a připojení stavu do ***AgentHandleru***, který je popsán níže.

```
import React from 'react';
import ReactDOM from 'react-dom';
import { createStore } from 'redux';
import { Provider } from 'react-redux';
import AgentHandler from './AgentHandler';
import injectScript from './injectScript';
import Validator from './redux/components';
import rootReducer from './redux/store';
import '../../style/main.scss';
const store = createStore(rootReducer);
new AgentHandler(store);

window.addEventListener('load', () => {
  ReactDOM.render(
    <Provider store={store}>
      <Validator />
    </Provider>,
    document.getElementById('container'),
  );
});
```

### 4.8.3 Web extension

#### 4.8.3.1 Manifest soubor

Klíčový soubor *manifest.json* pro rozšíření do prohlížeče vypadá takto.

```
{
  "browser_specific_settings": {
    "gecko": {
      "id": "martas.kristof@gmail.com",
      "strict_min_version": "48.0"
    }
  },
  "manifest_version": 2,
  "name": "BEM Validator",
  "author": "Martin Kristof",
  "description": "A DevTools extension for inspecting CSS and validate
for BEM",
  "version": "0.0.1",
  "content_security_policy": "script-src 'self'; object-src 'self'",
  "permissions": ["activeTab"],
  "devtools_page": "devtools.html",
  "background": {
    "scripts": ["background.js"],
    "persistent": false
  },
  "icons": {
    "16": "icon.png",
    "48": "icon.png",
    "128": "icon.png"
  },
  "content_scripts": [
    {
      "matches": ["<all_urls>"],
      "js": ["content-script.js"],
      "all_frames": true,
      "match_about_blank": true
    }
  ]
}
```

Obsahuje základní metadata a potřebná oprávnění. V případě této aplikace jde o povolení spouštění skriptu uvnitř balíčku.

Nezbytná je taky konfigurace *devtools*, protože se jedná o tento druh rozšíření. Nastavení *background* a *content_scripts* směruje zatím na neexistující soubory.

#### 4.8.3.2 Devtools.html

Strohý soubor *devtools.html* se základní HTML strukturou. V hlavičce je vložený JavaScriptový soubor, který dál převezme řízení.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <script src="devtools.js"></script>
  </head>
  <body></body>
</html>
```

#### 4.8.3.3 Devtools.js

*Devtools.js* je skript pro vytvoření panelu do nástrojů pro vývojáře (devtools), který nastavuje jméno a mapuje HTML soubor pro samotný panel. Za zmínku stojí první řádek, který ošetřuje kompatibilitu mezi jednotlivými prohlížeči (Chrome, Firefox či Microsoft EDGE, atd…)

```
window.browser = (() => window.msBrowser || window.browser ||
window.chrome)();

browser.devtools.panels.create('BEM validator', '', 'panel.html');
```

#### 4.8.3.4 Panel.html

Podobně málo obsáhlý soubor s definicí struktury panelu. Důležitý je jediný element s ***id*** „***container***“, kam bude renderována React aplikace.

Zdroj aplikace je nalinkovaný v hlavičce souboru (*ui.bundle.js*) a společně se souborem závislostí (*vendor.bundle.js*).

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <script src="build/vendor.bundle.js"></script>
    <script src="build/ui.bundle.js"></script>
  </head>
  <body>
    <div id="container"></div>
  </body>
</html>
```

#### 4.8.3.5 Background.js

Pro navázání komunikace mezi pozadím (*background.js* souborem) a obsahem je nutné otevřít spojení. Toto lze realizovat prostřednictvím vytvořených portů pro konkrétní panel prohlížeče.

Pro příjem zpráv z obsahového skriptu poslouží kód:

```
browser.runtime.onMessage.addListener((request, sender) => {
  if (sender.tab) {
    const tabId = sender.tab.id;
    if (tabId in connections) {
      connections[tabId].postMessage(request);
    } else {
      console.log('Tab not found in connection list.');
    }
  } else {
    console.log('sender.tab not defined.');
  }
  return true;
});
```

Opačným směrem je toto zajištěno následujícími řádky, kde je také řešena inicializace a zánik spojení:

```
browser.runtime.onConnect.addListener((port) => {
  port.onMessage.addListener((request) => {
    if (request.name === 'init') {
      connections[request.tabId] = port;

      port.onDisconnect.addListener(() => {
        delete connections[request.tabId];
      });

      return;
    }

    browser.tabs.sendMessage(request.tabId, {
      name: request.name,
      data: request.data,
    });
  });
});
```

K tomu, aby byla aplikace schopna reagovat na události prohlížeče, které provede uživatel (stránka byla načtena nebo znovu načtena), slouží jednoduchá logika, která tuto informaci pošle aplikaci:

```
browser.tabs.onUpdated.addListener((tabId, changeInfo) => {
  if (tabId in connections && changeInfo.status === 'complete') {
    connections[tabId].postMessage({
      name: 'reloaded',
    });
  } else if (tabId in connections && changeInfo.status === 'loading') {
    connections[tabId].postMessage({
      name: 'loading',
    });
  }
});
```

#### 4.8.3.6 Content-script.js

Velmi podobně vypadá soubor pro obsahový skript *content-script.js*. Umožňuje přijímat a odesílat zprávy.

Pro zprávy, které přichází z agenta, poslouží tento kód, kde je důležité filtrovat pouze zprávy pro tuto aplikaci:

```
window.addEventListener('message', (event) => {
  if (event.source !== window) {
    return;
  }
  const message = event.data;

  if (typeof message !== 'object' || message === null || message.source
!== 'bem-validator-agent') {
    return;
  }
  browser.runtime.sendMessage(message);
});
```

V případě obráceného směru, tedy pro zprávy přijímané z *background skriptu* a posílané do *agenta*, funguje následující funkce:

```
browser.runtime.onMessage.addListener((request) => {
  request.source = 'bem-validator-devtools';
  window.postMessage(request, '*');
});
```

Vždy je potřeba nastavit zdroj požadavku, aby byla druhá strana schopna identifikovat, odkud zpráva přichází.

### 4.8.4 Agent.js

***Agent*** třída v souboru *Agent.js* funguje jako prostředník mezi obsahovým skriptem a aplikací.

Pro usnadnění komunikace je vhodné si vytvořit pomocnou funkci pro posílání zpráv pro tuto vrstvu.

```
const sendMessage = (name, data) =>
  window.postMessage(
    {
      source: 'bem-validator-agent',
      name,
      data: data || {},
    },
    '*',
  );
```

U samotné třídy stojí za zmínku posílání dat z navštívené stránky prostřednictvím zprávy.

```
constructor() {
    this.handlers = {
      connect: () => sendMessage('connected'),
      error: (error) => sendMessage('showError', error),
      getData: () => Agent.sendData(),
    };

    this.initDevtoolsMessageListener();
  }

  static sendData() {
    return sendMessage('sendData', { html:
document.documentElement.innerHTML, bodyClass: document.body.className
});
  }
```

Jak vidno, třída při svém vzniku zaregistruje *listener* na zprávy, na které v různých případech reaguje jinak.

Nejdůležitější je dostat CSS třídy použité v HTML dále do další vrstvy pro validaci. Je zde ošetřen i výskyt CSS třídy přímo na *body tagu*.

### 4.8.5 AgentHandler.js

I pro vrstvu ***AgentHandler*** je vytvořena obdobná pomocná funkce pro posílání zpráv. Klíčová funkcionalita třídy je naslouchání na zprávy, které přicházejí z Agenta a reagování na ně.

Třída ve svém konstruktoru zaregistruje *listener* na zprávy, ke kterým definuje reakci.

V případě zprávy o připojení si zažádá o data z Agenta (CSS třídy použité v DOMU stránky). Při akci načítání zavolá akci pro zobrazení načítacího stavu aplikace.

Pokud uživatel změní URL nebo provede ***refresh*** stránky, znovu připojí aplikaci.

Když přijdou data prostřednictvím zprávy s klíčem „***sendData***“, provede volání metody ***getInvalidRules***, která pomocí dalšího modulu vyseparuje z HTML ony CSS třídy, v dalším kroku z nich pomocí knihovny ***css*** parsuje AST a ten předá knihovně *CSS-should-plugin-bem* přes metodu ***processLint***. V případě, že se vyskytne nějaká chyba, vyhodí akci pro zobrazení chyby a v opačném případě provede volání akce pro uložení vrácených dat do aplikace.

```
constructor(store) {
  this.store = store;

  port.onMessage.addListener((message) => {
    this.handleMessage(message);
  });

  this.handlers = {
    connected: () => sendMessage('getData'),
    loading: () => this.store.dispatch(showLoading()),
    reloaded: () => injectScript(),
    sendData: ({ html, bodyClass }) =>
this.store.dispatch(saveLintedRules(this.getInvalidRules(html,
bodyClass))),
    showError: (error) => this.store.dispatch(showError(error)),
  };
}

getInvalidRules(html, bodyClass) {
  try {
    const classes = extract(html, bodyClass);
    const parsedAst = parse(classes);

    return processLint(parsedAst);
  } catch (error) {
    this.handlers.showError(error.message);

    return {};
  }
```

### 4.8.6 Readme

Soubor *README.md* obsahuje základní informace o aplikaci jako jsou informace o autorovi, odkaz na repozitář a instrukce k instalaci a dalšímu vývoji.

Nedílnou součástí je seznam skriptů ke spuštění s popisem.

### 4.8.7 Changelog

Vzhledem k tomu, že je aplikace verzovaná dle sémantického verzování, je nezbytné jednotlivé změny k verzím zapisovat do souboru *CHANGELOG.md*, který pak poslouží jako historie změn.

### 4.8.8 CSS-should-plugin-bem

Hlavní logika validování (lintování) je umístěna právě v knihovně *CSS-should-plugin-bem*. Nutnou úpravou je rozšíření API knihovny.

Klíčovou funkcí, kterou lze zavolat je funkce ***processLint***, která přijímá jako parametr AST objekt daného CSS.

Tento objekt je dále předán do metody ***process***, kde dochází k přidání ***x-should*** deklarace pro detekované BEM třídy do předaného AST.

Strom je nakonec předán metodě ***lint***, která získá kolekci pravidel, která jsou nevalidní a také informaci o tom, jestli byl vůbec BEM zápis detekován.

```
export const lint = (css: Stylesheet): { rules: ILintRule[]; isValid:
boolean; isBemDetected: boolean } => {
  const { rules, isBemDetected } = getInvalidRules(css);

  return {
    isBemDetected,
    isValid: isBemDetected && rules.length === 0,
    rules: rules.map(({ rule, className }) => ({
      missingClassName: className,
      selector: rule.selectors[0] || '',
    })),
  };
};
```

Metoda ***getInvalidRules*** jen proiteruje všechna pravidla a volá na nich metodu ***detectInvalidRules***, která se stará o samotnou detekci chyb v zápisu a ukládání informací do příslušných kolekcí.

```
 if (isClassNameExistInCollection(invalidClassNames, selectors,
className)) {
          return;
        }

        if (BemParser.isBemModifier(selector) &&
BemParser.isBemBlock(className)) {
          const regex = new RegExp(className, 'g');
          const matched = selector.match(regex);

          if (matched.length <= 1) {
            invalidClassNames.push(className);
            invalidRules.push({ className, rule });

            return;
          }
        }

        if (BemParser.isBemBlock(selector) &&
!BemParser.isBemModifier(className) && !BemParser.isBemBlock(className))
{
          invalidClassNames.push(className);
          invalidRules.push({ className, rule });

          return;
        }

        if (BemParser.isBemModifier(selector)) {
          invalidClassNames.push(className);
          invalidRules.push({ className, rule });

          return;
        }
```

Zbytek knihovny prošel opravami, které byly nalezeny během TDD vývoje, kde nebyly ošetřeny různé okrajové případy (edge case).

Další nezbytnou úpravou byla konfigurace Webpacku pro umožnění buildování do výstupního souboru pro umožnění instalace přes NPM, kam je balíček publikován.

Mimo tyto změny došlo k přidání nástrojů pro zlepšení kvality kódu jako TSlint a Prettier.

## 4.9 Testování

Kromě zmíněných jednotkových testů je aplikace otestována i manuálně skupinou uživatelů z komunity vývojářů.

Testy pro instalaci a funkcionality doplňku byly provedeny v prohlížečích Firefox (65), Chrome (72), Opera (58), Vivaldi (2.3) a Brave (0.58.21) v jejich posledních verzích.

Vzhledem k dodržení specifikace *web extension API* se předpokládá, že bude doplněk funkční i v prohlížeči Chromium a Microsoft Edge. V Edge se rozšíření nainstalovalo, ale nezobrazil se panel v devtools. Pro plnou funkčnost v tomto prohlížeči by bylo zapotřebí přeportovat verzi přidáním speciální vrstvy[66].

## 4.10 Vytváření instalačního balíčku

K zabalení balíčku pro instalaci či publikaci lze využít knihovnu Web-ext. Tímto způsobem lze vytvořit z CLI archiv, který pak lze dále distribuovat pro Firefox.

V případě Chrome, Opera či Vivaldi nebo Brave se balíček dá vytvořit přímo přes prohlížeč na odpovídající URL pro správu rozšíření (např.: ***chrome://extensions/***) se zapnutou volnou režimu pro vývojáře. K tomu slouží tlačítko Zabalit rozšíření.

Akce vygeneruje soubor formátu „***crx***“, který pak lze instalovat prostřednictvím stejné URL adresy. Soubor se jednoduše přemístí do prohlížeče funkcí drag & drop a povolí se instalace.

## 4.11 Publikace do AMO

Balíček je připraven k publikaci. Po vytvoření vývojářského účtu na ***https://addons.mozilla.org/*** lze nahrát konkrétní verzi rozšíření do repozitáře.

Tato akce se obvykle vykonává přes prohlížeč a odkaz „Přidání nového doplňku“, kde je následně nabídnuto nahrání souboru zip.

V případě distribuce v AMO lze pak balíček najít v prohlížeči na stránce Správy rozšíření ***about:addons*** ve vyhledávacím poli.

## 4.12 Publikace do NPM

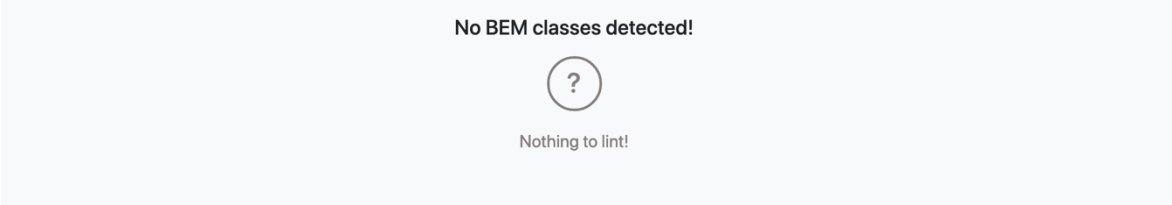
Samotný doplněk na NPM publikován nebude, nicméně knihovna *CSS-should-plugin-bem* už publikována je.

Došlo k tedy k vypuštění nových verzí po změnách, které byly zmíněny výše.

[66] Portování rozšíření do EDGE verze https://docs.microsoft.com/en-us/microsoft-edge/extensions/guides/porting-chrome-extensions

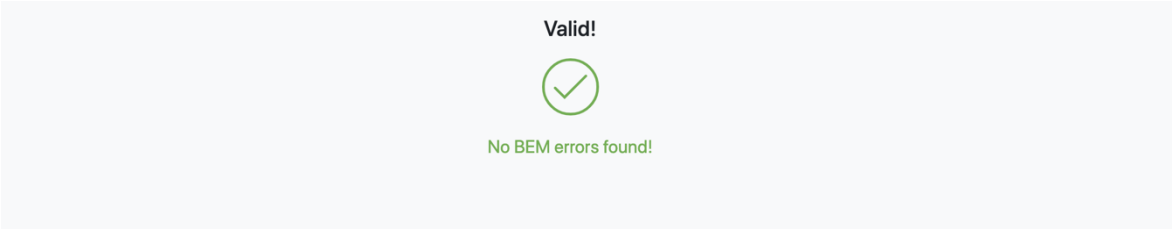
## 4.13 Ukázka aplikace

### *4.13.1 Stav při nedetekování BEMu na stránce*



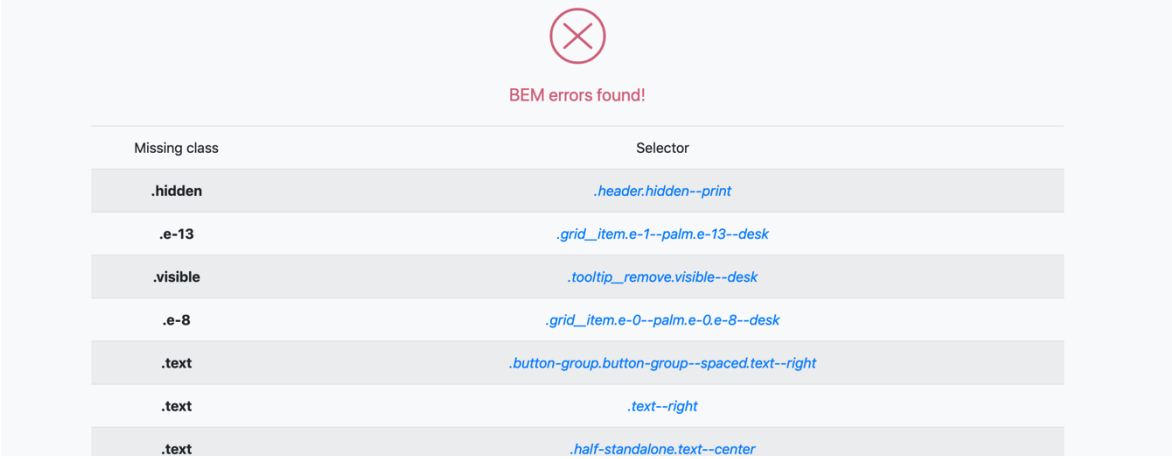

**Obrázek 27: Ukázka aplikace při nedetekování BEMu na stránce (zdroj: vlastní)**

### *4.13.2 Stav při validním zápisu dle BEMu*




**Obrázek 28: Ukázka aplikace při validním stavu (zdroj: vlastní)**

### *4.13.3 Stav při nevalidním zápisu dle BEMu*




**Obrázek 29: Ukázka aplikace při nevalidním stavu (zdroj: vlastní)**

#### 4.13.3.1 Detail nevalidního selektoru po kliknutí na jeho název


**Obrázek 30: Detail nevalidního selektoru, v tomto případě chybí třída *"text"* (zdroj: vlastní)**

# 5 Výsledky a diskuse

Aplikace byla navržena, vyvinuta a otestována. Splňuje akceptační kritéria. Nakonec byla publikována do AMO repozitáře a je volně dostupná pro instalaci.

Dále lze rozšíření instalovat a plně používat v dalších prohlížečích Opera, Chrome, Chromium, Vivaldi a Brave. Dosavadní testování neodhalilo chyby, nicméně v případě nalezení chyb je možno tuto skutečnost reportovat na Github stránce projektu.

Dokončením této práce byl splněn návrh z mojí bakalářské práce, kde v diskuzi tento nápad zazněl.

Prostor pro vylepšení je zejména v možnosti konfigurace pro zápis oddělovače bloku a modifikátoru BEMu. Toto vylepšení není součástí této závěrečné práce, ale pevně doufám, že se k němu dostanu. Aplikace by se také dala lokalizovat do více jazyků.

Publikace doplňků by mohla být dokončena i pro ostatní prohlížeče, přičemž u Chrome to je velmi reálné v blízké době.

Výsledný doplněk nebylo možno porovnat s konkurenčními nástroji, protože v této formě neexistují. Jedinou konkurencí jsou nástroje při vývoji, a to například pluginy pro PostCSS jako post-css-bem-linter[67].

Výhodou mého doplňku do prohlížeče je zejména to, že jej lze spouštět jak na produkčních webech, tak i na vývojových bez znalosti konkrétního kódu. Rychlost renderování je záležitostí Reactu.

Aplikace neumí bohužel odhalit špatně sémanticky pojmenované prvky – například když programátor pojmenuje třídu odstavce jako tlačítko apod.

Během psaní celé práce nabral spoustu zkušeností s psaním tohoto typu aplikace, protože je to moje první tvorba rozšíření do prohlížeče. Také jsem si mohl vyzkoušet nové technologie.

Pro tvorbu diagramů byl použit online nástroj draw.io[68]. Pro kreslení wireframů jsem využil aplikace Balsamiq[69]. Ke psaní kódu bylo použito IDE PHPStorm[70] od Jetbrains.

[67] Odkaz na PostCSS plugin https://github.com/postcss/postcss-bem-linter
[68] Online aplikace Draw.io https://www.draw.io/
[69] Web aplikace Balsamiq https://balsamiq.com/
[70] Odkaz na IDE PHPStorm https://www.jetbrains.com/phpstorm/

# 6 Závěr

Cílem závěrečné práce bylo navrhnout a vytvořit rozšíření do prohlížeče pro kontrolu zápisu tříd kaskádových stylů dle metodiky BEM. Tato část byla splněna v praktické části.

Výsledná aplikace je publikována v repozitáři doplňků pro Mozilla Firefox, případně ji lze nainstalovat přímo do ostatních prohlížečů ze souboru. ***BEM Validator*** je dostupný pod MIT licencí na Githubu a AMO a je také součástí přílohy k závěrečné práci.

V části teoretických východisek byly splněny i dílčí cíle, a to nejen popsat použité technologie, ale také informace o dostupných prohlížečích a o tom, jak BEM funguje a jaké jsou charakteristiky Web extensions. Konkrétněji jsou popsány knihovny React, Redux a další použité nástroje.

Definován je také postup publikování do výše zmíněných zdrojů, kde je aplikace dostupná veřejnosti.

V diskuzi jsem zhodnotil funkčnost aplikace a nastínil další budoucnost projektu.

# 7 Seznam použitých zdrojů


**CZ.NIC. 2014.** Prohlížeče a internetové technologie. *Jak na internet.* [Online] CZ.nic, 2014. https://www.jaknainternet.cz/page/1235/prohlizece-a-internetove-technologie/.

**Abernathy, Christine. 2015.** Integrating Parse and React Native for iOS. *RayWenderlich.com.* [Online] 4. 8 2015. https://www.raywenderlich.com/1729-integrating-parse-and-react-native-for-ios.

**Abramov, Dan. 2019.** Ecosystem. *Redux.* [Online] 2019. https://redux.js.org/introduction/ecosystem.

**—. 2019.** Motivation. *Redux.* [Online] 2019. https://redux.js.org/introduction/motivation.

**—. 2019.** Redux. *Redux.* [Online] 2019. https://redux.js.org/.

**—. 2019.** Three Principles. *Redux.* [Online] 2019. https://redux.js.org/introduction/three-principles.

**—. 2019.** Usage with React. *Redux.* [Online] 2019. https://redux.js.org/basics/usage-with-react.

**Bootstrap team. 2019.** Bootstrap. *Bootstrap.* [Online] 2019. https://getbootstrap.com/.

**Bos, Bert.** CSS. *W3C.* [Online] [Citace: 13. 12 2016.] https://www.w3.org/Style/CSS/ .

**Browser Extension Community Group . 2017.** Browser Extensions. *Browser Extension Community Group* . [Online] Browser Extension Community Group , 23. červenec 2017. https://browserext.github.io/browserext/.

**Dev.Opera. 2019.** Publishing Guidelines. *Dev.Opera.* [Online] 2019. https://dev.opera.com/extensions/publishing-guidelines/.

**—. 2019.** Testing and Debugging. *Dev.Opera.* [Online] 2019. https://dev.opera.com/extensions/testing/.

**Facebook Inc. 2019.** Components and Props. *React.* [Online] 2019. https://reactjs.org/docs/components-and-props.html.

**—. 2018.** Getting Started. *React.* [Online] 2018. https://reactjs.org/docs/getting-started.html.

**—. 2019.** Handling Events. *React.* [Online] 2019. https://reactjs.org/docs/handling-events.html.

**—. 2019.** Higher-Order Components. *React.* [Online] 2019. https://reactjs.org/docs/higher-order-components.html.

**—. 2019.** React. *React.* [Online] 2019. https://reactjs.org.

**—. 2019.** React. *Introducing JSX.* [Online] 2019. https://reactjs.org/docs/introducing-jsx.html.

**—. 2019.** Rendering Elements. *React.* [Online] 2019. https://reactjs.org/docs/rendering-elements.html.

**—. 2019.** State and Lifecycle. *React.* [Online] 2019. https://reactjs.org/docs/state-and-lifecycle.html.

**Gimeno, Alberto. 2018.** Testing with Jest: from zero to hero. *LogRocket.* [Online] 30. 11 2018. https://blog.logrocket.com/testing-with-jest-from-zero-to-hero-85ce0e9cc953.

**Google Chrome. 2018.** Content Scripts. *Google Chrome.* [Online] 2018. https://developer.chrome.com/extensions/content_scripts.

**—. 2018.** Debugging Extensions. *Google Chrome.* [Online] 2018. https://developer.chrome.com/extensions/tut_debugging.

**—. 2018.** Extending DevTools. *Google Chrome.* [Online] 2018. https://developer.chrome.com/extensions/devtools.

**—. 2018.** Message Passing. *Google Chrome.* [Online] 2018. https://developer.chrome.com/extensions/messaging.

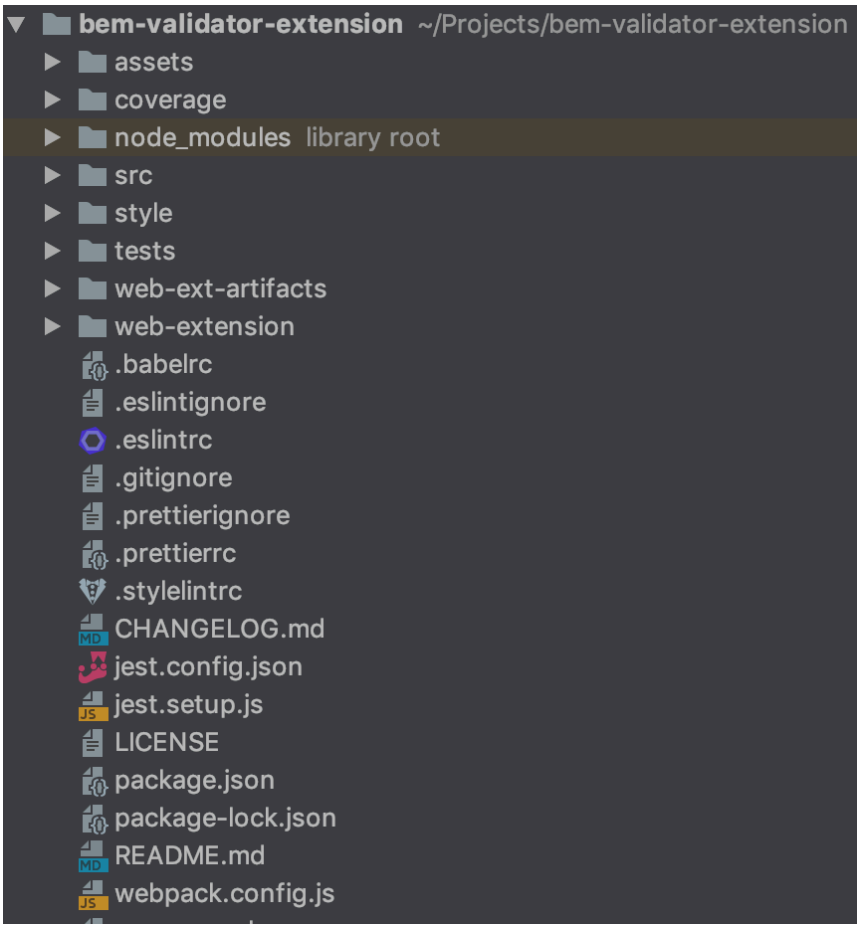
**—. 2019.** Publish in the Chrome Web Store. *Google Chrome.* [Online] 2019. https://developer.chrome.com/webstore/publish.
**—. 2018.** What are extensions? *Google Chrome.* [Online] 2018. https://developer.chrome.com/extensions.
**IT Slovník. 2017.** Linting. *IT Slovník.* [Online] 2017. [Citace: 19. 2 2017.] https://it-slovnik.cz/pojem/linting.
**JS Foundation. 2019.** About. *ESLint.* [Online] 2019. https://eslint.org/docs/about/.
**Krištof, Martin. 2017.** *Nástroj pro validaci zápisu kaskádových stylů.* Katedra informačního inženýrství, Česká zemědělská univerzita v Praze. Praha : Česká zemědělská univerzita v Praze, Provozně ekonomická fakulta, Katedra informačního inženýrství, 2017. Bakalářská práce.
**Michálek, Martin. 2017.** https://www.vzhurudolu.cz/prirucka/bem. *Vzhůru dolů.* [Online] 5. 6 2017. [Citace: 20. 1 2019.] https://www.vzhurudolu.cz/prirucka/bem.
**—. 2018.** Stylelint, protože pořádek musí být. I v CSS kódu. *Vzůru dolů.* [Online] 21. 5 2018. https://www.vzhurudolu.cz/prirucka/stylelint.
**Mozilla. 2019.** Content scripts . *Mozilla.* [Online] 21. 1 2019. https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Content_scripts.
**—. 2018.** Content Security Policy . *Mozilla.* [Online] 3. 6 2018. https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Content_Security_Policy.
**—. 2019.** Getting started with web-ext . *Mozilla.* [Online] 3. 3 2019. https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Getting_started_with_web-ext.
**—. 2018.** What are extensions? . *MDN web docs.* [Online] 2018. https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/What_are_WebExtensions.
**—. 2017.** Working with files . *Mozilla.* [Online] 30. 11 2017. https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Working_with_files.
**—. 2018.** Anatomy of an extension . *MDN web docs.* [Online] 2018. https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Anatomy_of_a_WebExtension.
**—. 2019.** Debugging. *Mozilla.* [Online] 3. 3 2019. https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Debugging.
**—. 2019.** JavaScript APIs . *Mozilla.* [Online] 19. 2 2019. https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API.
**—. 2019.** manifest.json . *Mozilla.* [Online] 18. 1 2019. https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/manifest.json.
**—. 2019.** Permissions. *Mozilla.* [Online] 22. 2 2019. https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/manifest.json/permissions.
**—. 2018.** Sideloading add-ons . *Mozilla.* [Online] 16. 11 2018. https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Distribution_options/Sideloading_add-ons#Preparing_your_add-on.
**—. 2018.** Submitting an add-on . *Mozilla.* [Online] 15. 2 2018. https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/Distribution/Submitting_an_add-on.
**—. 2018.** WebExtensions. *MDN web docs.* [Online] 2018. https://developer.mozilla.org/cs/docs/Mozilla/Add-ons/WebExtensions.

**npmjs.** Publishing npm packages. *docs.npmjs.com.* [Online] [Citace: 8. 2 2017.] https://docs.npmjs.com/getting-started/publishing-npm-packages#publishing-the-package.
**Reactstrap. 2018.** Reactstrap. *Reactstrap.* [Online] 2018. https://reactstrap.github.io/.
**Tahir, Nish. 2018.** Lessons Learned Implementing Redux on Android. *Hackernoon.* [Online] 19. 2 2018. https://hackernoon.com/lessons-learned-implementing-redux-on-android-cba1bed40c41.
**Wikipedia. 2018.** Syntaktický strom. *Wikipedia.* [Online] 5. 9 2018. https://cs.wikipedia.org/wiki/Syntaktick%C3%BD_strom.

# 8 Přílohy

Na CD je přiložen zdrojový kód aplikace. Postup ke zprovoznění aplikace je popsán v ***README.md*** souboru v kořenovém adresáři projektu.

Struktura adresářů je zobrazena na obrázku:




**Obrázek 31: Struktura adresářů projektu (zdroj: vlastní)**