Pedagogická fakulta
Faculty of Education

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích
Pedagogická fakulta
Katedra informatiky

Bakalářská práce

CSS preprocesory

Vypracoval: Martin Jinda
Vedoucí práce: PaedDr. Petr Pexa, Ph.D.

České Budějovice 2014

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH
Fakulta pedagogická
Akademický rok: **2013/2014**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Martin JINDA**
Osobní číslo:	**P11046**
Studijní program:	**B7507 Specializace v pedagogice**
Studijní obor:	**Informační technologie a e-learning**
Název tématu:	**CSS preprocesory**
Zadávající katedra:	**Katedra informatiky**

Zásady pro vypracování:

CSS preprocesory umožňují dynamické programování v jazyce DSL (Dynamic StyleSheet language), který se více blíží klasickému programování v porovnání s editací klasických CSS. Výhodou oproti standardnímu zápisu stylových předpisů je přínos v objektově orientovaném zápisu kódu, možnosti vytváření proměnných pro snadné modifikace hodnot, které se vyskytují na více místech v stylových předpisech, ošetření prefixovaných / neprefixovaných vlastností a je možné také používat matematické binární operace. CSS preprocesor je také nástroj, který z daného kódu zapsaného ve vlastní syntaxi vygeneruje CSS předpis pro webové stránky. Tato nová technologie začíná být mezi pokročilými webovými vývojáři velmi vyhledávána především pro práci na velkých webových projektech. Mezi nejznámější preprocesory patří SASS, LESS a Stylus. Cílem bakalářské práce je zpracovat problematiku CSS preprocesorů při tvorbě webového rozhraní a vytvořit internetový manuál, ve kterém bude popsáno jak s preprocesory pracovat, jejich instalace, pro jaké webové projekty se nejvíce hodí a jak efektivně vytvářet webové stránky pomocí těchto velmi užitečných nástrojů. Dále bude provedeno porovnání syntaxe a výhod a nevýhod všech zvolených preprocesorů nejen mezi sebou, ale i v porovnání s klasickými CSS. Praktickou část bakalářské práce bude tvořit webová aplikace, realizovaná s využitím CSS preprocesorů, na níž bude technologie detailně prakticky představena. Součástí práce bude i dotazníkové šetření zaměřené na zjištění aktuálního rozšíření CSS preprocesorů mezi webovými vývojáři a firmami zabývajícími se tvorbou webových stránek.

Rozsah grafických prací: **CD ROM**

Rozsah pracovní zprávy: **40**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

1. SASS. Sass: Syntactically Awesome Style Sheets [online]. 2006 [cit. 2014-06-03]. Dostupné z: http://www.sass-lang.com/
2. LESS. Less [online]. 2009 [cit. 2014-06-03]. Dostupné z: http://www.lesscss.org/
3. STYLUS. Stylus - Expressive, dynamic, robust CSS [online]. 2009 [cit. 2014-06-03]. Dostupné z: http://learnboost.github.io/stylus/
4. W3C. Cascading Style Sheets [online]. 1994 [cit. 2014-06-03]. Dostupné z: http://www.w3.org/Style/CSS/
5. MICHÁLEK, Martin. Průvodce CSS preprocesory [online]. 2014 [cit. 2014-06-03]. Dostupné z: http://www.vzhurudolu.cz/blog/12-css-preprocesory-1
6. GOOGLE. Working with CSS Preprocessors [online]. 2014 [cit. 2014-06-03]. Dostupné z: https://developer.chrome.com/devtools/docs/css-preprocessors
7. PAINTER, Bermon. GOOGLE. CSS Pre-Processors: Stylus, Less & Sass [online]. 2014 [cit. 2014-06-03]. Dostupné z: https://speakerdeck.com/bermonpainter/css-pre-processors-stylus-less-and-sass

Vedoucí bakalářské práce: **PaedDr. Petr Poxa, Ph.D.**
Katedra informatiky

Datum zadání bakalářské práce: **9. června 2014**

Termín odevzdání bakalářské práce: **30. dubna 2015**

Mgr. Michal Vančura, Ph.D.
děkan

L.S.

doc. PaedDr. Jiří Vaníček, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 16. dubna 2013

Prohlášení

Prohlašuji, že svoji bakalářskou jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské – diplomové – rigorózní – disertační práce, a to v nezkrácené podobě – v úpravě vzniklé vypuštěním vyznačených částí archivovaných … fakultou elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne 2. prosince 2014.

Podpis studenta

Abstrakt

Cílem bakalářské práce je zpracovat a popsat problematiku CSS preprocesorů při tvorbě webového rozhraní. V rámci práce představím, jak efektivně vytvořit webové stránky pomocí těchto velmi užitečných nástrojů, jež preprocesory obsahují. CSS preprocesory umožňují dynamické programování, jedná se o dynamický jazyk DSL (Dynamic Stylesheet language) který se blíží více programování. Výhodou oproti standardnímu psaní stylových předpisů je přínos v objektově orientovaném zápisu kódu. Zde lze používat také matematické binární operace (násobení, dělení, sčítání a odčítání).

V rámci práce bude vytvořen dotazníkový průzkum zaměřený na aktuální rozšíření CSS preprocesorů mezi internetové agentury zabývající se tvorbou webových stránek a odbornou veřejností v České republice.

Klíčová slova

CSS, preprocesor, webová stránka, LESS, SASS, Stylus

Abstract

The aim of this thesis is to process and describe technology of CSS preprocessors in developing a web interface. Within my thesis I am going to present how to effectively create a website using these very useful tools which contains preprocessors. CSS preprocessors allows dynamic programming, this is a dynamic language DSL (Dynamic Stylesheet language), this close to more programming. The advantage beside the standard writing style rule is the benefit in writing object-oriented code. It is possible to use also binary math operations (multiplication, division, addition and subtraction)

Within my thesis I will created questionnaire focused on the current extension CSS preprocessors to internet agencies deal with developing websites and professional community in Czech Republic.

Keywords

CSS, preprocesor, web page, LESS, SASS, Stylus

Poděkování

Rád bych touto formou poděkoval vedoucímu mé závěrečné práce panu PaedDr. Petru Pexovi, Ph.D. za odborné vedení při zpracování práce, cenné rady, ochotu a trpělivost při vedení mé práce.

Obsah

1 Úvod

CSS preprocesory umožňují dynamické programování v jazyce DSL (Dynamic StyleSheet language), který se více blíží klasickému programování v porovnání s editací klasických CSS. Výhodou oproti standardnímu zápisu stylových předpisů je přínos v objektově orientovaném zápisu kódu, možnosti vytváření proměnných pro snadné modifikace hodnot, které se vyskytují na více místech v stylových předpisech, ošetření prefixovaných / neprefixovaných vlastností a je možné také používat matematické binární operace. CSS preprocesor je také nástroj, který z daného kódu zapsaného ve vlastní syntaxi vygeneruje CSS předpis pro webové stránky. Tato nová technologie začíná být mezi pokročilými webovými vývojáři velmi vyhledávána především pro práci na velkých webových projektech. Mezi nejznámější preprocesory patří SASS, LESS a Stylus.

Cílem bakalářské práce je zpracovat problematiku CSS preprocesorů při tvorbě webového rozhraní a vytvořit internetový manuál, ve kterém bude popsáno jak s preprocesory pracovat, jejich instalace, pro jaké webové projekty se nejvíce hodí a jak efektivně vytvářet webové stránky pomocí těchto velmi užitečných nástrojů. Dále bude provedeno porovnání syntaxe a výhod a nevýhod všech zvolených preprocesorů nejen mezi sebou, ale i v porovnání s klasickými CSS.

Praktickou část bakalářské práce bude tvořit webová aplikace, realizovaná s využitím CSS preprocesorů, na níž bude technologie detailně prakticky představena. Součástí práce bude i dotazníkové šetření zaměřené na zjištění aktuálního rozšíření CSS preprocesorů mezi webovými vývojáři a firmami zabývajícími se tvorbou webových stránek.

1.1 Cíle práce

Cílem práce je dopodrobna rozebrat a popsat všechny možnosti nejčastěji používaných CSS preprocesorů, konkrétně se jedná o SASS, LESS a Stylus. Zjistit všechny možnosti práce s preprocesory, popsat co vlastně preprocesory jsou, pro jaké webové projekty se nejvíce hodí a porovnat výhody a nevýhody CSS preprocesorů s obyčejným CSS a jaké výhody to přinese webovým vývojářům.

Dále vytvořím webovou stránku, na které bude uveden postup krok po kroku, jak s preprocesory začít, jejich samotná instalace, srovnání syntaxe a přehledný manuál pro lidi kteří budou chtít s touto technologií začít.

Součástí práce bude proveden dotazníkový průzkum o rozšířenosti CSS preprocesorů mezi odbornou veřejností v České republice. Dotazník bude zpracován pomocí několika otázek, ten poté rozešlu mezi webové vývojáře na volné noze a několika firmám zabývajícím se tvorbou webových projektů na internetu. Z výsledku dotazníku zjistím, jak moc je technologie preprocesorů využívaná a rozšířená v České republice. Cílem bude rozšířit tuto technologii a usnadnit začátek vytváření komplexního webového projektu.

1.2 Východiska práce

V dnešní době velkých webových projektů a kooperace mnoha lidí na takovém projektu, je třeba používat různá vylepšení, která usnadní práci a dohledávání chyb v kódu. Při běžném psaní CSS předpisů může soubor obsahovat až přes 2.000 řádků. Jakmile v takovém stylopisu potřebujete něco operativně změnit, je třeba vynaložit značné úsilí, aby kód zůstal čitelný a srozumitelný pro ostatní spolupracovníky. Většinu těchto problému vyřeší CSS preprocesor, nástroj který Vám z daného kódu, napsaný ve vlastní syntaxi, vygeneruje validní CSS, funkční ve všech dnes již běžně používaných internetových prohlížečích.

Samotné CSS pro profesionální použití některé věci neumí např. vytváření proměnných pro snadné modifikace hodnot, které se vyskytují na více místech v stylových předpisech, ošetření prefixovaných / neprefixovaných vlastností, vnořené definice a binární operace (sčítání, odčítání, násobení a dělení).

Tato technologie začíná být mezi pokročilými webovými vývojáři velmi vyhledávaná pro její snadné použití.

1.3 Metody práce

Vytvořím seznam několika otázek, z nichž posléze vytvořím dotazník, který následně rozešlu do několika firem zabývající se tvorbou webových prezentací a mezi

odbornou veřejnost, po určité době zpracuji odpovědi rozšířeností technologie v České republice. Dotazník bude elektronický a bude vytvořen pomocí webové služby Google Form.

V úvodu práce proberu novinky ve tvorbě webových stránek. Jaké jsou možnosti při tvorbě webových stránek, aktuální trendy a postupy.

Poté přejdu na samotnou práci s CSS preprocesory, budu testovat jejich výhody a nevýhody, syntaxi, instalaci a kompilaci kódu. Po otestování všech preprocesorů a jejich zdokumentování vytvořím samotnou internetovou prezentaci, kde uvedu veškeré potřebné informace, které jsem při práci zjistil.

2 HTML

První definice jazyka HTML, celým názvem HyperText Markup Language, byla vytvořena v roce 1991 jako součást projektu WWW[1], který měl umožnit vědcům pracujícím v CERNu[2], umožnit zpřístupnění vědeckých textů v počítačové síti. Tato verze HTML byla popsána v dokumentu HTML Tags[3]. Umožňovala text rozčlenit do určitých logických úrovní, použít několik druhů zvýraznění textu a zařadit do textu odkazy a obrázky. Při navrhování autor nepředpokládal znalost daného jazyka. První verze byla psána přímo pro operační systém NextStep a obsahovala jak prohlížeč, tak i integrovaný editor. V roce 1992 byl veřejně uvolněn jazyk HTML.

Aktuální verze HTML5, která by měl být schválená v posledním čtvrtletí roku 2014, přináší podstatné změny v technologiích webových stránek, umožňuje kromě jiného přehrávat multimediální obsah přímo ve webovém prohlížeči a vytvářet v něm vlastní aplikace, které fungují i bez připojení k internetové síti. [1]

2.1 XHTML

Před vydáním HTML5 se samotné HTML 4.01 nevyvíjelo a přišel pokus o XHTML. XHTML je stejný značkovací jazyk jako HTML, které vychází ze standardu pro výměnu dat XML. XHTML oproti HTML nic nového nepřináší, žádné nové možnosti a specifikaci, ale jen omezení.

Rozlišujeme tři druhy XHTML:

- **XHTML 1.0 Strict** – Čistě strukturovaný dokument, neobsahuje žádné značky s formátováním vzhledu (např. font, b, i, u).
- **XHTML 1.0 Transitional** – Povoluje atributy pro formátování textu a odkazů v elementu body a některé další atributy.

[1] World Wide Web – světová rozsáhlá síť neboli celosvětová síť
[2] Centre Européenne pour la Recherche Nucléaire, Evropské centrum jaderného výzkumu
[3] http://www.w3.org/History/19921103-hypertext/hypertext/WWW/MarkUp/Tags.html

- **XHTML 1.0 Frameset** – Používá se při použití rámců na webu pro rozdělení okna prohlížeče na dvě nebo více částí.

Rozdíl mezi HTML a XHTML je v zápisu daného kódu stránky. XHTML kód musí být správně strukturovaný, tagy musí být psány malýma znaky. Každý tag je párový, jestliže dřívější tag byl nepárový musí se před poslední špičatou závorku zapsat lomítko.

Příklad: `<img src="obrazek.jpg" title="Obrázek" />` místo `<img src="obrazek.jpg" title="Obrázek">`.

Element html vždy obsahuje dva elementy, head (hlavičku) a body (tělo dokumentu). Hlavička navíc musí povinně obsahovat element title.

2.2 DHTML

Jako DHTML, neboli Dynamické HTML, se označuje spojení HTML, JavaScriptu, kaskádových stylů a někdy DOM[4]. Toto spojení zavedla společnost Microsoft ve svém prohlížeči Internet Explorer 4. Nejedná se tedy o samostatný jazyk nebo jeho novou verzi, jde o soubor nástrojů a postupů kombinující ostatní technologie. DHTML umožňuje skriptovacím jazykům změnu proměnné v definici webové stránky, to ovlivní vzhled a funkce jinak statické stránky i poté co je kompletně načtená. Tyto změny jsou navíc vázány na konkrétní akce návštěvníka (např. kliknutí na určité místo na webu, přejetí myší přes daný prvek, …). [2]

[4] Document Object Model – objektový model dokumentu. Objektově orientovaná reprezentace XML nebo XHTML dokumentu.

3 CSS

3.1 Co je CSS

CSS neboli Cascading Style Sheets, v překladu kaskádové styly. Kaskádové protože při psaní CSS souboru se využívá takzvaných kaskád, to znamená, že se na sebe mohou vrstvit definice stylu, ale platí jen ta poslední. Jedná se o jazyk, který určuje způsob zobrazení elementů v souborech napsaných ve značkovacích jazycích HTML nebo XHTML. Jazyk navrhla standardizační organizace W3C[5]. Hlavní myšlenkou CSS je umožnit vývojářům oddělit vzhled daného dokumentu od jeho struktury a obsahu. Soubor s kaskádovými styly má vždy příponu css. Samotný soubor k ničemu neslouží, funkčnost stylů se projeví až po připojení s daným HTML souborem. Jedním CSS souborem lze zajistit stejný vzhled, různým HTML souborům.

3.2 CSS 1

První specifikace jazyka CSS vyšla jako doporučení mezinárodního konsorcia W3C v prosinci roku 1996. CSS 1 obsahovalo hlavně specifikace fontů, barvy písma, pozadní elementů, parametry písma, zarovnání elementů, definování rámečků, vnitřní a vnější okraje elementů. Obsahovalo také rozdělení třídy a identifikátory, pro jednoznačnou identifikaci elementů v dokumentu.

3.3 CSS 2

Rok 1998 dal vzniknout nové specifikaci CSS 2, kde se již daly formátovat dokumenty XHTML i XML. Začal rozlišovat výstup pro monitor, tiskárnu, čtecí zařízení či mobil. Přibylo také absolutní a relativní pozicování.

[5] World Wide Web Consortium – mezinárodní konsorcium, jeho členové společně s veřejností vyvíjejí webové standardy.

3.4 CSS 2.1

CSS 2.1 vzniklo jako reakce na problémy s CSS 2, které obsahovalo velké množství chyb ve specifikaci. Vyšlo v roce 2002, odstranilo chyby z předchozí verze, vypustilo specifikaci vlastností, které nebyly v prohlížečích implementovány, a přidávalo vlastnosti, které prohlížeče samy implementovaly, bez předchozího souhlasu konsorcia W3C.

3.5 CSS 3

Aktuálně nejnovější verze jazyka CSS, která se vyvíjí již od roku 2005. Očekávané dokončení se předpokládá na rok 2015, ale již dnes je většina vlastností, které obsahuje podporována nejběžnějšími webovými prohlížeči. Mezi nové vlastnosti patři např. zaoblené rohy, stín u blokového prvku, stín u textu, transformace objektů a další.

3.6 Připojení CSS stylů k HTML, XHTML

Připojení CSS stylů může probíhat několika možnými způsoby, v praxi se nejčastěji používá odkaz na externí soubor.

3.6.1 Přímé připojení stylu k prvku HTML

Do určitého prvku, na který chceme aplikovat určité CSS vlastnosti, vložíme atribut style, vlastnost a její hodnotu.

Příklad:

```
<h1 style="color: red; letter-spacing: 2px;">Nadpis první úrovně</h1>
```

3.6.2 Vložení pomocí stylopisu do HTML dokumentu

Stylopis je seznam stylů pro daný dokument zapsaný mezi tag `<style type="text/css"></style>` v hlavičce dokumentu. Vhodné pro použití tam, kde chceme použít určitý styl jen pro jeden dokument.

Příklad:

```
<style type="text/css"> h2 { color: red; letter-spacing: 2px; } </style>
```

3.6.3 Připojením externího souboru *.css

Připojení probíhá pomocí speciálního odkazu, opět v hlavičce dokumentu. Tento způsob je jediný možný pokud váš web obsahuje více jak dva html dokumenty, zápisem zajistíte, že každý stránka bude mít stejné stylové předpisy. Změna jednoho CSS souboru se poté projeví na všech stránkách.

Příklad:

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

nebo

```
<style type="text/css">@import url("css/styl.css");</style>
```

4 CSS preprocesory

CSS preprocesor je nástroj, který z vámi napsaného zdrojového kódu, vygeneruje výsledný, validní kaskádový styl. Preprocesor je pro každého kdo dělá na jakékoliv webové stránce, umožňuje zjednodušit a zrychlit psaní přehledného CSS kódu. Pokud na webovém projektu spolupracuje více kodéru nebo programátorů, může kód stylů nabývat přes tisíce řádků, ve kterých se velmi těžko orientuje a provádí jakékoliv zásadní změny. Pomocí technologie preprocesorů zvýšíte přehlednost a udržovatelnost vašeho kódu, která je při vývoji velmi důležitá. V klasickém psaní kaskádových stylů nemůžeme využívat vlastnosti, jako jsou proměnné, vnořování kódu, mixiny (znovu použitelné bloky kódu), podmínky, cykly a další. Díky CSS preprocesorů jsou všechny tyto prvky dostupné. Preprocesorů je celá řada, každý má vlastní syntaxi, vlastní kompilaci. V této práci se budu věnovat, v dnešní době nejpoužívanějším CSS preprocesorům LESS, SASS a Stylus.

4.1 Srovnání dostupných preprocesorů

CSS preprocesorů je celá řada, jedná se vlastně o parser, který je nastavený pro jeho vlastního syntaxi, kterou následně kompiluje do validního CSS zápisu. Většina z dříve dostupných preprocesorů již na internetu nenajdeme, ty ještě dostupné jsou velmi dlouho neaktualizované a obsahují jen ty nejzákladnější funkce. [3]

Název	Webová dokumentace	Jazyk	Verze	Komunita	Posl. aktualizace
LESS	Ano	Javascript	2.1.0	11 442	23. 11. 2014
SASS	Ano	Ruby	3.4.8	4 885	14. 11. 2014
Stylus	Ano	Javascript	0.49.3	5 045	6. 11. 2014
Rework	Ano	Javascript	1.0.1	1 819	20. 5. 2014
Roole	Ano	Javascript	0.9.0	380	2013
PCSS	Ano	PHP	0.71	-	-
CSS Crush	Ano	PHP	2.2.0	440	2013
HitchJS	Ano	Javascript	0.6.3	168	2102

Switch CSS	Ne	Python	-	-	25. 6. 2007
Turbine	Ne	PHP	-	-	-
CSS Cacheer	Ne	PHP	-	-	2008
Dt CSS	Ano	PHP	-	-	4. 2. 21010
CSS PP	Ne	PHP, Python	-	-	-
Pornel	Ano	PHP	-	-	15. 11. 2007
Spiffing	Ano	PHP	-	138	2011
CSScaffold	Ano	PHP	-	20	2010

Tabulka č. 1 - Srovnání dostupných css preprocesorů část 1

Název	**Pokročilé funkce**	**Podpora CSS3**	**Dostupnost**	**Možnost kompilace**
LESS	Ano	Ano	Ano	Terminál, GUI, IDE, Online v prohlížeči
SASS	Ano	Ano	Ano	Terminál, GUI, IDE
Stylus	Ano	Ano	Ano	Terminál, GUI, IDE
Rework	Ne	Ano	Ano	Terminál
Roole	Ne	Ano	Ano	Terminálem
PCSS	Ne	Ne	Ano	PHP script
CSS Crush	Ne	Ano	Ano	PHP skriptem
HitchJS	Ne	Ne	Ano	Javascript
Switch CSS	Ne	Ne	Ano	-
Turbine	-	-	-	-
CSS Cacheer	Ne	Ne	Ne	-
Dt CSS	Ne	Ne	Ano	PHP script
CSS PP	Ne	Ne	Ne	-
Pornel	Ne	Ne	Ano	PHP script
Spiffing	Ne	Ne	Ano	PHP script
CSScaffold	Ne	Ne	Ano	PHP script

Tabulka č. 2 - Srovnání dostupných css preprocesorů část 2

Vysvětlení k srovnávacím tabulkám:

- **Webová dokumentace** – Má webovou dokumentaci

- **Jazyk** – V jakém programovacím jazykem je napsaný
- **Verze** – Aktuální vydaná verze
- **Komunita** – Počet unikátních hodnocení na jeho repositáři
- **Poslední aktualizace** – Datum poslední aktualizace
- **Pokročilé funkce** – Obsahuje pokročilé funkce (podmínky, cykly, extend, url(), objekty, definice vlastních funkcí, ...)
- **Podpora CSS3** – Podporuje poslední standard syntaxe CSS 3
- **Dostupnost** – Je preprocesor dostupný volně z internetu
- **Možnost kompilace** – Jaké možnosti nabízí pro kompilaci (Terminálem / příkazovou řádkou, GUI – Speciální aplikací třetích stran, IDE – Plugin pro nějaký IDE editor, PHP skriptem)

5 LESS

LESS neboli Leaner CSS, v překladu štíhlejší CSS, je dynamic stylesheet language, jeho tvůrce je Alexis Sellier, známý také jako cloudhead[6]. Jedná se o open-source projekt. Všechny jeho zdrojové kódy jsou dostupné v repositářích na serveru Github.com[7]. První verze byla napsána v jazyce Ruby[8], v novějších verzích byl nahrazen jazyk Ruby jazykem Javascript. LESS rozšiřuje CSS jazyk, přidává možnosti, které umožňují například proměnné, mixiny, funkce a mnoho dalších technik, které si dopodrobna ukážeme dále. Tyto techniky slouží k tomu, aby se kód napsaný v CSS stal více udržitelným a rozšiřitelným. LESS běží jak na straně serverové, pomocí javascriptové, serverové knihovny Node nebo, tak i na klientské v internetovém prohlížeči. [4]

LESS funguje uvnitř javascriptové knihovny s názvem Node. Existuje ale i mnoho nástrojů třetích stran, které umožňují kompilaci a změny souborů mimo prostředí Node.

5.1 Jak používat LESS

Preprocesor LESS lze použít třemi způsoby. V příkazové řádce, pomocí npm (node package manager), stažením knihovny scriptu less.js, přímo v internetovém prohlížeči nebo existuje spousty nástrojů třetích stran pro nejsnadnější správu vašich LESS stylesheetů.

5.1.1 Použití pomocí příkazové řádky

K instalaci je potřeba mít nainstalovaný node package manager, který najdeme na oficiálních stránkách nodejs.org. Instalace pak probíhá velmi jednoduše, pomocí jednoho řádku. Instalace probíhá jen jednou.

```
$ npm install -g less
```

[6] http://www.cloudhead.io/ - Osobní web tvůrce CSS preprocesoru LESS.
[7] Github.com je server pro hosting open-source projektů verzovaných pomocí Git.
[8] Ruby

Jakmile máme LESS nainstalovaný, můžeme okamžitě začít kompilovat naše vytvořené soubory less. Kompilace opět probíhá jedním příkazem na jeden řádek.

```
$ lessc styles.less
```

Tento příkaz nám zkompiluje náš zápis do univerzálního souboru stdout[9]. Musíme proto na konec přidat ještě název výsledného souboru, do kterého se má less zkompilovat.

```
$ lessc styles.less > styles.css
```

Často se využívá také tzv. minimalizování kódu, to je odstranění mezer a řádků. Tato minimalizace vede k menší velikosti daného souboru a rychlejšímu načítání webové stránky. Slouží k tomu přepínač –x, zapsaný za lesscs.

5.1.2 Použití na straně klienta (prohlížeče)

LESS jako jediný preprocesor umí překládat stylopis přímo v internetovém prohlížeči pomocí javascriptu. Potřebujeme tedy nalinkovat soubor less.js do našeho webu a můžeme rovnou připojit soubory stylopisu less, zápis vypadá takto.

```
<link rel="stylesheet/less" type="text/css"
href="styles.less" />

<script src="less.js" type="text/javascript"></script>
```

Možnosti jsou definovány nastavením před zápisem vložení less.js v kódu.

```
<script>
  less = {
    env: "development",
    async: false,
    fileAsync: false,
    poll: 1000,
    functions: {},
    dumpLineNumbers: "comments",
    relativeUrls: false,
```

[9] stdout – Standardní výstup je proud, na který program zapisuje svá výstupní data.

```
    rootpath: ":/a.com/"
  };
</script>
<script src="less.js"></script>
```

5.1.2.1 Sledovací režim

Sledovací režim je vlastnost na klientské straně, umožňuje automaticky načíst styly v případě, že se změnily. Pro zapnutí této funkce je nutné přidat za adresu webové stránky, výraz `#!watch` a znovu načíst stránku. Téhož docílíme zavoláním `less.watch()` z konzole. [5]

5.1.3 Použití aplikací třetích stran

Aplikace třetích stran umožňují vyhnout se úplně jakékoliv práci s příkazovým řádkem, sledovacím režimem v prohlížeči apod. Stačí stáhnout a nainstalovat aplikaci, která tyto veškeré starosti vyřeší za vás. Seznam aplikací najdete na stránkách http://lesscss.org/usage/index.html#guis-for-less. Stejně se dá použít i online kompilátory na webu.

Pokud používáte pokročilý IDE[10] editor např. Sublime Text, Atom, Eclipse, VisualStudio, NetBeans a další, je vhodné, využít nástroje pro ně připravených, které se o kompilaci postarají za vás.

5.2 Proměnné

V programování je proměnná označení pro určitou hodnotu tzv. identifikátor, který uchovává určitou hodnotu nebo informaci. V css preprocesorech fungují proměnné velmi podobně jako v jiných programovacích jazycích. Umožní vám specifikovat často používanou hodnotu na jednom místě a následně jí použít v dalších sekcích stylového předpisu. Pokud změníme hodnotu v dané proměnné, změna se následně projeví v celém dokumentu. Usnadní to tak práci při dohledávání změn a přehlednější správu dokumentu. Počet proměnných v jednom dokumentu není

[10] IDE – Integrated Development Environment – vývojové prostředí

omezený. Hodnota může být například barva, definice velikosti písma a definici fontu. Proměnné lze také využívat i ve výpočtech.

V LESS zápis proměnných začíná klíčovým znakem @. Standardní zápis pak může vypadat takto:

```
@barva: #FF0000;

@fixedWidth: 890px;

#block {

     color: @barva;

     width: @fixedWidth;

}
```

Výsledný, zkompilovaný css soubor:

```
#block {
    color: #FF0000;
    width: 890px;
}
```

5.3 Vnořená pravidla

Vnořená pravidla slouží k vnořování elementů pod jejich rodičovské elementy. Dosáhneme tak standardního kaskádového zápisu. Zápis je velmi jednoduchý a jedná se o nejvíce používanou funkci v CSS preprocesorech.

Zápis v LESS:

```
#header {

  color: black;


  .navigation {

    font-size: 12px;

  }

  .logo {
```

```
    width: 300px;
    &:hover { text-decoration: none }
  }
}
```

Výsledné CSS:

```
#header { color: black; }
#header .navigation {
  font-size: 12px;
}
#header .logo {
  width: 300px;
}
#header .logo:hover {
  text-decoration: none;
}
```

5.4 Mixiny

Mixiny umožňují vložit předem definované vlastnosti třídy do jiné třídy, specifikací jejího názvu uvnitř jiné třídy. Mixiny lze využívat jako funkce a pracovat s argumenty, podobně jako v programovacích jazycích. Mixiny můžou být neparametrické nebo parametrické.

Neparametrické mixiny slouží již ke zmíněnému vkládání sad vlastností z jedné sady do pravidel sady jiné. Lze kombinovat i například s proměnnými. Jako mixin můžeme použít jakoukoliv class (třídu) nebo id (identifikátor) se sadou definic.

Příklad zápisu mixinu:

```
@modra: #0000FF;
.ohraniceni {
```

```
    border-bottom: 1px solid @modra;
    border-left: 1px solid red;
    border-right: 1px solid red;
}
#menu {
    .ohraniceni;
}
```

Výsledný CSS soubor:

```
.ohraniceni {
    border-bottom: 1px solid #0000FF;
    border-left: 1px solid red;
    border-right: 1px solid red;
}
#menu {
    border-bottom: 1px solid #0000FF;
    border-left: 1px solid red;
    border-right: 1px solid red;
}
```

5.4.1 Parametrické mixiny

Parametrické mixiny se používají stejně jako neparametrické mixiny, s výjimkou toho, že dovolují použití parametrů. Nejčastější využití najdeme v zápisu prefixů například pro vlastnosti zaobleného ohraničení border-radius. Kde je potřeba ošetřit zápis pro všechny prohlížeče, jednoduchým zápisem pak zajistíte funkčnost napříč všemi prohlížeči.

Zápis v LESS:

```
.border-radius (@radius) {
    border-radius: @radius;
    -moz-border-radius: @radius;
    -webkit-border-radius: @radius;
```

```
}
```

Zde použití mixin pro .border-radius, který obsahuje prefixy k zajištění funkčnosti ve všech běžně používaných prohlížečích. K němu patří parametr @radius, do kterého v dalšímu zápisu vložíme hodnotu.

Zápis v LESS:

```
.tlacitko {
  .border-radius(10px);
}
```

Výsledný CSS soubor:

```
.tlacitko {
     -webkit-border-radius: 10px;
     -moz-border-radius: 10px;
     -ms-border-radius: 10px;
     border-radius: 10px;
}
```

Parametrické mixiny mohou mít již přednastavené výchozí hodnoty pro své parametry. Zápis pak vypadá takto:

```
.border-radius (@radius: 10px) {
  border-radius: @radius;
  -moz-border-radius: @radius;
  -webkit-border-radius: @radius;
}
```

5.4.2 Mixiny s @argumenty

@argumenty mají v mixinech speciální význam, obsahují všechny předané argumenty v okamžiku, kde je mixin volán. Lze toho využít v případě, kdy nechcete pracovat s jednotlivými parametry. Více na příkladu.

Zápis v LESS:

```
.box-shadow (@x: 0, @y: 0, @blur: 1px, @color: #000) {
  box-shadow: @arguments;
  -moz-box-shadow: @arguments;
  -webkit-box-shadow: @arguments;
}
.box-shadow(2px, 5px);
```

Výstup v CSS:

```
box-shadow: 2px 5px 1px #000;
  -moz-box-shadow: 2px 5px 1px #000;
  -webkit-box-shadow: 2px 5px 1px #000;
```

5.4.3 Media query pomocí mixinů

Media query mohou být vnořená stejným způsobem jako selektory. Přepínače jsou zkopírovány do těla dotazu médií.

Zápis v LESS:

```
.screencolor{
  @media screen {
    color: green;
    @media (min-width:768px) {
    color: red;
    }
    }
  @media tv {
    color: black;
  }
}
```

Výstup v CSS:

```
@media screen {
  .screencolor {
    color: green;
  }
}
@media screen and (min-width: 768px) {
  .screencolor {
    color: red;
  }
}
@media tv {
  .screencolor {
    color: black;
  }
}
```

5.4.4 Mixiny jako funkce

Proměnné a mixiny definované v mixinech jsou viditelné, a mohou být použity v rozsahu volajícího. Existuje pouze jedna výjimka, proměnnou nelze kopírovat, pokud volající obsahuje proměnnou se stejným názvem (který zahrnuje proměnné definované jiným mixin volání). Pouze proměnné, které jsou přítomné v aktuálně volaném mixinu jsou chráněny. Proměnné zděděné z mateřských oborů mají přednost.

Zápis v LESS:

```
.mixin() {
  @width:  100%;
  @height: 200px;
```

```
}
.caller {
  .mixin();
  width:  @width;
  height: @height;
}
```

Výstup v CSS:

```
.caller {
  width:  100%;
  height: 200px;
}
```

Tyto proměnné definované v mixin mohou fungovat jako její návratové hodnoty. To nám umožňuje vytvořit mixin, který může být použit téměř jako funkce. Příklad.

Zápis v LESS:

```
.average(@x, @y) {
  @average: ((@x + @y) / 2);
}
div {
  .average(16px, 50px); // "call" the mixin
  padding: @average;    // use its "return" value
}
```

Výsledné CSS:

```
div {
  padding: 33px;
}
```

5.5 Práce se vzorky a výrazy

V mixinu lze změnit chování v závislosti na parametru, který mu předáváme. Přednastavíme si mixin aby při zavolání light nebo dark, zadaný parametr zesvětlil nebo ztmavil.

```
.mixin (dark, @color) {
  color: darken(@color, 10%);
}
.mixin (light, @color) {
  color: lighten(@color, 10%);
}
.mixin (@_, @color) {
  display: block;
}
```

Pokud nyní do proměnné uložíme light nebo dark. Barva se nám bude zesvětlovat nebo ztmavovat, podle zavolané funkce.

```
@switch: light;
.class {
  .mixin(@switch, #888);
}
```

Výsledné CSS:

```
.class {
  color: #a2a2a2;
  display: block;
}
```

5.6 Mixin strážci

Strážce slouží v případě, kdy chceme párovat na základě výrazů, na rozdíl od jednoduchých hodnot nebo arity. Ve snaze zůstat co nejvíce k deklarativní povaze CSS, rozhodl se tvůrce LESS, spouštění podmíněných mixinů pomocí hlídacích mixinů, tzv. strážcích, místo konstrukcí if / else ve stylu specifikací @media query.

Klíčový výraz je zde when, který zavádí posloupnost strážců. Příklad:

```
.mixin (@a) when (lightness(@a) >= 50%) {
  background-color: black;
}
.mixin (@a) when (lightness(@a) < 50%) {
  background-color: white;
}
.mixin (@a) {
  color: @a;
}

.class1 { .mixin(#ddd) }
.class2 { .mixin(#555) }
```

Výsledné CSS:

```
.class1 {
  background-color: black;
  color: #ddd;
}
.class2 {
  background-color: white;
  color: #555;
}
```

Kompletní výčet porovnávacích operátorů pro použití se strážci je tento: > >= = =< <. Klíčové slovo true je jediná kladná hodnota, z čehož plyne, že následující dva mixiny jsou ekvivalentní:

```
.truth (@a) when (@a) { ... }

.truth (@a) when (@a = true) { ... }
```

Jakákoliv jiná hodnota než klíčové slovo true je záporná.

```
.class {
  .truth(40);
}
```

Strážci mohou být odděleni čárkou ',' pokud je kterýkoli ze strážců vyhodnocen jako kladný, je považován za vyhovujícího:

```
.mixin (@a) when (@a > 10), (@a < -10) { ... }
```

Argumenty můžete rovněž porovnávat mezi sebou nebo s ne argumenty:

```
@media: mobile;

.mixin (@a) when (@media = mobile) { ... }

.mixin (@a) when (@media = desktop) { ... }


.max (@a, @b) when (@a > @b) { width: @a }

.max (@a, @b) when (@a < @b) { width: @b }
```

Chcete-li porovnávat mixiny podle typu hodnoty, můžete použít funkci `is`. Veškeré další funkce, které můžete využít k porovnávání jsou v následující kapitole.

```
.mixin (@a, @b: 0) when (isnumber(@b)) { ... }

.mixin (@a, @b: black) when (iscolor(@b)) { ... }
```

Lze použít také klíčová slova and, vytvoří tak další podmínky uvnitř strážce a not, k negaci podmínky.

```
.mixin (@a) when (isnumber(@a)) and (@a > 0) { ... }

.mixin (@b) when not (@b > 0) { ... }
```

5.7 Vestavěné funkce

Seznam a příklady podporovaných funkcí v LESS.

5.7.1 Práce s barvami

LESS poskytuje sadu funkcí pro převody barev. Všechny barvy jsou převáděny do formátu HSL[11] a následně se s nimi manipuluje na úrovni kanálů.

```
rgb(@r, @g, @b);
// převede na barvu v hexadecimálním tvaru
rgba(@r, @g, @b, @a);
// převede na barvu argb(@color); argb(@color);
// převede na barvu v hexadecimálním tvaru - #AARRGGBB
hsl(@hue, @saturation, @lightness);
// převede na RGB v hexadecimálním tvaru
hsla(@hue, @saturation, @lightness, @alpha);
// vytvoří barvu z modelu HSLA
hsv(@hue, @saturation, @value);
// převede na RGB v hexadecimálném tvaru
hsva(@hue, @saturation, @value, @alpha);
// vytvoří bravu z modelu HSVA

hue(@color);        // vrátí odstín barvy
saturation(@color); // vrátí nasycení barvy
lightness(@color);  // vrátí světlost barvy
red(@color);        // vrátí červenou složku
green(@color);      // vrátí zelenou složku
blue(@color);       // vrátí modrou složku
```

[11] HSL (Hue, Saturation, Lightness) – barevný model

```
alpha(@color);      // vrátí alpha kanál barvy
luma(@color);       // vrátí černobílou složku

saturate(@color, 10%); // zvýší nasycení barvy o 10%
desaturate(@color, 10%); // sníží nasycení barvy o 10%
lighten(@color, 10%); // zesvětlí barvu o 10%
darken(@color, 10%); // ztmaví barvu o 10%
fadein(@color, 10%); // sníží průhlednost barvy o 10%
fadeout(@color, 10%); // svýší průhlednost barvy o 10%
fade(@color, 50%); // nastaví průhlednost barvy na 50%
spin(@color, 10); // Posune odstín o 10 stupňů doprava
mix(@color1, @color2, [@weight: 50%]);  // vrátí mix dvou barev
greyscale(@color); // vrátí šedou barvu
contrast(@color1,  [@darkcolor:  black],  [@lightcolor: white], [@threshold: 43%]);
// vrátí @darkcolor pokud v @color1 bude > 43% černobílé složky, jinak vrátí @lightcolor

multiply(@color1, @color2);
// porovná korespondující RGB kanály, které spojí a vrátí tmavší barvu
screen(@color1, @color2);
// porovná korespondující RGB kanály, které spojí a vrátí světlejší barvu
overlay(@color1, @color2);
```

```
// kombinuje efekty multiply() a screen(). Podmíněně vrátí světlé kanály světlejší a tmavé tmavší. Výsledek určuje první barva

softlight(@color1, @color2);

// Kombinace dvou barev, kde color1 je světlá barva a color2 barva která bude zesvětlena.

hardlight(@color1, @color2);

// obdobná funkce jako overlay(), color1 slouží k překrytí barvě druhé. Na základě toho bude color2 buďto světlejší nebo tmavší

difference(@color1, @color2);

exclusion(@color1, @color2);

average(@color1, @color2);

negation(@color1, @color2);
```

5.7.2 Další funkce

```
escape(@string);

// zákóduje řetězec pro URL

e(@string);

// escapuje textový řetězec

%(@string, values...);

// zformátuje řetězec (např. %("Vysledek: %d", 5))


unit(@dimension, [@unit: ""]);

// odstraní nebo změní jednotku rozměru

color(@string); // parsuje řetězec na barvu
```

5.7.3 Logické hodnoty

```
isnumber(); // vrací true, pokud je parametr číslo
```

```
isstring(); // opak funkce isnumber(). Pokud je parametr string, vrátí true

iscolor(); // vrací true, pokud je parametr barva

isurl(); // vráci true pokud je parametr url(...)

ispixel(); // vrací true pokud je parametr v pixelech

isem(); // vrací true pokud je parametr v em jednotkách

ispercentage(); // vrací true pokud je parametr v procentech

isunit(11px, px); // vrací true, pokud se shoduje jednotka se zadanou hodnotou
```

5.7.4 Matematické funkce

```
sin(); // Převede úhel na goniometrickou funkci sinus

asin(); // Převede goniometrickou funkci sinus na úhel

cos(); // Převede úhel na goniometrickou funkci cosinus

acos(); // Převede goniometrickou funkci cosinus na úhel

tan(); // Převede uhél na goniometrickou funkci tangent

atan(); // Převede goniometrickou funkci tangent na úhel

pi(); // Vrátí číslo π (pi)

min(); // Vrátí nejmenší hodnotu ze zadaných

max(); // Vrátí maximální hodnotu ze zadaných

ceil(@number); // zaokrouhlí číslo nahoru

floor(@number); // zaokrouhlí číslo dolů

percentage(@number); // převede číslo na procenta (0.8 -> 80%)

round(number, [places: 0]); // zaokrouhlí číslo
```

5.8 Převod jednotek

První argument obsahuje číslo s jednotkou a druhý argument obsahuje jednotky, do kterých chceme první argument převést. Pokud jsou oba argumenty kompatibilní, číslo bude převedeno, pokud ne, vrátí se první argument nezměněn.

Kompatibilní jednotky:

- délka: m, cm, mm, in, pt a pc
- čas: s a ms
- úhel: rad, deg, grad a turn

```
convert(@number, "unit"); // převod jednotek.
```

5.9 Výpočty

Jakékoli číslo, barva nebo proměnná mohou být součástí nějakého výpočtu. LESS rozpozná rozdíly mezi definicemi barev a jednotkami. Příklad:

```
@var: 1px + 5;
```

LESS rozpoznal, že první hodnota má definici jednotky v px a proto druhou, bez jednotky, bude počítat také jako px. Výsledek vrátí 6px.

Další příklady výpočtů:

```
@base: 5%;

@filler: @base * 2;

@other: @base + @filler;


color: #888 / 4;

background-color: @base-color + #111;

height: 100% / 2 + @filler;
```

5.10 Komentáře

V zápis LESS lze využít dva druhy komentářů, první druh komentáře je víceřádkový, který se objeví v následném CSS souboru. Druhý druh je jednořádkový, jeho obsah se v kompilovaném CSS neobjeví.

```
/* První druh komentáře */

// Druhý druh komentáře
```

5.11 Import

Do nezkompilovaného souboru LESS, lze importovat jak soubor LESS, tak soubor CSS. U souboru typu CSS, je nutné použít koncovku *.css, typ souboru LESS, může, ale neumí využít danou koncovku *.less. Zápis pak vypadá takto:

```
@import "lib.less";

@import "lib";

@import "lib.css";
```

Všechny proměnné a mixiny použité v importovaných souborech, budou dostupné hlavnímu soubor. Pokud importujeme preprocesorový zápis, import v less nebude fungovat, tak jak v CSS, kde vytváří http requesty navíc, ale vloží daný, napsaný kód z importovaného souboru přímo do souboru. Pokud chcete aby se css soubor choval stejně jako less , upravte importující příkaz takto:

```
@import (less) "fancybox.css";
```

6 SASS

SASS (Syntactically Awesome StyleSheets), vznikl roku 2006 a jeho hlavními tvůrci jsou Hampton Catlin, Natalie Weizenbaum, Chris Eppstein je css preprocesor, který umožňuje použití nadstandardních funkcí, které klasické css neumí. Mezi tyto funkce patří např. proměnné, mixiny , jmenné prostory a další, to vše se syntaxí plně kompatibilní s CSS3. SASS je stejně jako LESS plně open-source projekt pod licencí MIT, veškeré jeho zdrojové kódy jsou dostupné ke stažení v repositářích GitHub na adrese https://github.com/sass/sass.

6.1 Syntaxe

K dispozici jsou dva druhy zápisu pro SASS. První, známí jako SCSS (Sassy CSS), tento zápis je nejvíce podobný standardnímu zápisu CSS, využívá složené závorky a středníky. Soubory s tímto zápisem mají koncovku *.scss.

Druhý, a starší zápis syntaxe je SASS, poskytuje stručnější zápis. Místo uzavření složenou závorkou se v zápisu používá odsazení a nové řádky místo středníků. Obě syntaxe mají stejné vlastnosti i když některé funkce mají odlišný zápis, tyto soubory mají koncovku *.sass.

Veškeré příkladové kódy budou psány v syntaxi SCSS, u více rozlišených zápisu uvedu i zápis syntaxí SASS.

Syntaxe lze mezi sebou převádět pomocí vnitřního nástroje Sass-convert, ten je dostupný z příkazové řádky.

Konverze SASS do SCSS

```
$ sass-convert style.sass style.scss
```

Konverze SCSS do SASS

```
$ sass-convert style.scss style.sass
```

6.2 Jak používat SASS

SASS je napsaný v jazyce Ruby, proto i instalace probíhá pomocí tohoto jazyka. SASS lze používat třemi způsoby, jako nástroj příkazového řádku, samotný Ruby modul nebo pomocí aplikace třetích stran.

6.2.1 Použití pomocí příkazové řádky

Instalace probíhá velmi jednoduše, jedním příkazem zadaným do terminálového okna resp. příkazové řádky. Pokud budete SASS instalovat na systém Windows, musíte nejdříve nainstalovat komponentu Ruby.

```
gem install sass
```

Nyní byste měli mít preprocesor SASS nainstalovaný, ověřit si to můžete tímto příkazem:

```
sass –v
```

Pokud instalace proběhla úspěšně, zobrazí se Vám tato zpráva:

```
Sass 3.4.7 (Selective Steve)
```

Pro spuštění jednorázové kompilace slouží příkaz:

```
sass vstupscss.scss vystupcss.css
```

nebo

```
sass vstupsass.sass vystupcss.css
```

Můžete také říct, aby SASS sledoval souboru a aktualizoval CSS pokaždé změně, stačí přidat `--watch`

```
sass --watch vstupscss.scss: vystupcss.css
```

nebo

```
sass --watch vstupsass.sass: vystupcss.css
```

nebo

6.2.2 Použití pomocí jazyka Ruby

Použití SASS v prostředí Ruby je více komplikované. Musí se zde kompilátor (SASS::Engine) vyvolávat ručně, zadáním následujících příkazů.

```
engine = Sass::Engine.new("#main {background-color: #0000ff}", :syntax => :scss)
engine.render #=> "#main { background-color: #0000ff; }\n"
```

Tento způsob slouží hlavně pro vytváření nových, vlastních pluginů, které budou pracovat pod SASS. Více informací najdete v oficiální dokumentaci.

6.2.3 Použití pomocí aplikací třetích stran

Seznam nejlepší aplikací placených i neplacených naleznete na oficiálních stránkách SASS, http://sass-lang.com/install.

Pokud používáte pokročilý IDE editor např. Sublime Text, Atom, Eclipse, VisualStudio, NetBeans a další, je vhodné, využít nástroje pro ně připravených, které se o kompilaci postarají za vás.

6.3 Proměnné

Do proměnných se dají uložit libovolné hodnoty, nejčastěji jde o barvy, rodinu písem, velikost odsazení a další. V SASS definice proměnné začíná znakem `$`, následuje název proměnné a její hodnota, oddělené dvojtečkou.

Zápis v SCSS:

```
$brandcolor: #FF0000;
$fixedWidth: 890px;

#block {
    color: $brandcolor;
    width: $fixedWidth;
}
```

Výsledné CSS:

```
#block {
    color: #FF0000;
```

```
    width: 890px;
}
```

Pokud definujeme proměnnou uvnitř nějakého selektoru, platí daná proměnné pouze pro daný selektor. Chceme-li aby byla globální pro celý stylopis, přidáme za danou proměnnou `!global`.

Zápis v SCSS:

```
#main {
  $width: 5em !global;
  width: $width;
}
#sidebar {
  width: $width;
}
```

Výsledné CSS:

```
#main {
  width: 5em;
}

#sidebar {
  width: 5em;
}
```

6.4 Vnořená pravidla

Vnořená pravidla slouží stejně jako v LESS k vytváření hierarchie v daném CSS kódu. Mějte na paměti, že čím více vnořených elementů, tím více specifikovaný daná css třída.

Zápis v SCSS:

```
nav {
  ul {
    margin: 0;
    padding: 0;
    list-style: none;
  }
  li { display: inline-block; }
  a {
    display: block;
    padding: 6px 12px;
    text-decoration: none;
  }
}
```

Výsledné CSS:

```
nav ul {
  margin: 0;
  padding: 0;
  list-style: none;
}
nav li {
  display: inline-block;
}
nav a {
  display: block;
  padding: 6px 12px;
```

```
  text-decoration: none;
}
```

Pokud dva a více selektorů mají stejné vlastnosti, můžeme využít komplexnější zápis vnořených pravidel.

Zápis v SCSS:

```
#main {
  width: 97%;

  p, div {
    font-size: 2em;
    a { font-weight: bold; }
  }

  pre { font-size: 3em; }
}
```

Výsledné CSS:

```
#main {
  width: 97%; }
  #main p, #main div {
    font-size: 2em; }
    #main p a, #main div a {
      font-weight: bold; }
  #main pre {
    font-size: 3em; }
```

Zápis pseudo tříd probíhá použití znaku & před danou pseudo třídu. Nejčastější využití najdeme u odkazů, kde chceme nastavit pomocí CSS jak se bude chovat při najetí kurzoru myši.

Zápis v SCSS:

```
a {
  font-weight: bold;
  text-decoration: none;
  &:hover { text-decoration: underline; }}
```

Výsledné CSS:

```
a {
  font-weight: bold;
  text-decoration: none; }
  a:hover {
    text-decoration: underline; }
```

Znak & primárně slouží k přenesení rodičovského elementu v rámci vnořeného pravidla.

Zápis v SCSS:

```
#main {
  color: black;
  &-sidebar { border: 1px solid; }
}
```

Výsledné CSS:

```
#main {
  color: black; }
  #main-sidebar {
    border: 1px solid; }
```

6.5 Import

Vždy je vhodné rozdělit si stylopis do několika menších, parciálních komponent, které pak importujeme do hlavního stylopisu. Import se v SASS chová stejně jako v LESS, nevytváří import i ve výsledném CSS, ale kopíruje kód parciálních do finální. [6]

Importované komponenty musejí mít před názvem _, aby import rozpoznal, co se kam má importovat. Příklad:

Máme-li napsaný _reset.scss, který chceme vložit do finálního stylopisu screen.scss:

```
// _reset.scss
html,
body,
ul,
ol {
  margin: 0;
  padding: 0;
}

// screen.scss
@import 'reset';

body {
  font: 100% Helvetica, sans-serif;
  background-color: #efefef;
}
```

V příkazu nemusí uvádět, o kterou syntaxi se jedná, SASS si přebere oba kódy.

Výsledné CSS:

```
html, body, ul, ol {
```

```
  margin: 0;
  padding: 0;
}
body {
  font: 100% Helvetica, sans-serif;
  background-color: #efefef;
}
```

6.6 Mixiny

Mixiny slouží ke vkládání kódu, který se často opakuje, nejčastěji používaný k definicím prefixových hodnot pro funkčnost napříč prohlížeči. Na příkladu si vytvoříme mixin s názvem large-text, který bude mít nastavený větší velikost a tučnost. Mixin se v selektoru volá pomocí `@include`.

Zápis v SCSS:

```
@mixin large-text {
  font: {
    family: Arial;
    size: 20px;
    weight: bold;
  }
  color: #ff0000;
}
.page-title {
  @include large-text;
  padding: 4px;
  margin-top: 10px;
```

```
}
```

Výsledné CSS:

```
.page-title {
  font-family: Arial;
  font-size: 20px;
  font-weight: bold;
  color: #ff0000;
  padding: 4px;
  margin-top: 10px; }
```

6.6.1 Parametrické mixiny

Parametrické mixiny obsahují navíc parametr, který můžeme měnit pro každý element zvlášť.

Zápis v SCSS:

```
@mixin border-radius($radius) {
  -webkit-border-radius: $radius;
     -moz-border-radius: $radius;
      -ms-border-radius: $radius;
          border-radius: $radius;
}
.box { @include border-radius(10px); }
```

Zápis v SASS:

```
=border-radius($radius)
  -webkit-border-radius: $radius
  -moz-border-radius:    $radius
  -ms-border-radius:     $radius
  border-radius:         $radius
```

```
.box
  +border-radius(10px)
```

Vytvořili jsme si mixin s názvem border-radius, který nám ošetří prefixový zápis pro `border-radius`, do parametru `$radius` poté vložíme hodnotu, kterou má `border-radius` u všech zápisů nabývat. Mixin lze použít vícekrát.

Výsledné CSS:

```
.box {
  -webkit-border-radius: 10px;
  -moz-border-radius: 10px;
  -ms-border-radius: 10px;
  border-radius: 10px;
}
```

6.7 Jmenné prostory

CSS má poměrně málo vlastností, které jsou ve jmenných prostorách. Jedná se třeba o definici font, kde se v jedné řádce dá nastavit jak barva, rodina fontů, velikost, tučnost a další. To vše nahradí jednotlivé vlastnosti např. color, font-family, font-size atd. V preprocesorech si můžete jednoduše tyto jmenné prostory vytvářet a nebo využívat těch existujících.

Zápis v SCSS:

```
.funky {
  font: {
    family: fantasy;
    size: 30em;
    weight: bold;
  }
```

```
}
```

Výsledné CSS:

```
.funky {
  font-family: fantasy;
  font-size: 30em;
  font-weight: bold; }
```

6.8 Extend

Funkce @extend je obsažena v preprocesoru SASS a Stylus, jedná se o jednu z nejvíce užitečných funkcí. Pomocí extend můžete snadno sdílet sadu CSS vlastností z jednoho selektorů do druhého. To pomáhá udržet váš zápis v SASS strohý. Příklad:

Zápis v SCSS:

```
.message {
  border: 1px solid #ccc;
  padding: 10px;
  color: #333;
}
.success {
  @extend .message;
  border-color: green;
}
.error {
  @extend .message;
  border-color: red;
}
.warning {
```

```
  @extend .message;
  border-color: yellow;
}
```

Výsledné CSS:

```
.message, .success, .error, .warning {
  border: 1px solid #cccccc;
  padding: 10px;
  color: #333;
}
.success {
  border-color: green;
}
.error {
  border-color: red;
}
.warning {
  border-color: yellow;
}
```

6.9 Komentáře

SASS podporuje víceřádkové komentáře zapsané mezi /* */ a jednořádkové komentáře //. Víceřádkové komentáře se nepromítnou do výsledného, zkompilovaného CSS souboru.

6.10 Funkce v SASS

Většina funkcí je shodná s LESS, například pro práci s barvami nebo matematické funkce, proto tyto obory zde uvádět nebudu.

6.10.1 Funkce pro práci s řetězci string

```
unquote($string)    // Odstraní quote z řetězce

quote($string)      // Přidá quote

str-length($string) // Vrátí počet znaků

str-insert($string,, $insert, $index)  // Vloží $insert do $string v $index

str-index($string, $substring)    // Vrátí index prvního výskutu o $substring v $string

str-slice($string, $start-at, [$end-at])    // Extrahuje podřetěze od $string

to-upper-case($string)  // Převede řetězec na velká písmena

to-lower-case($string)  // Převede řetězec na malá písmena
```

6.10.2 Funkce pro práci s List

```
Length($list)  // Vrátí velikost $list

nth($list, $n) // Vrátí n-tý člen v $list

set-nth($list, $n, $value)   // Nahradí n-tý člen v $list

join($list1, $list2, [$separator]) // Spojí dva listy do jednoho

append($list, $val, [$separator]) // Vloží hodnotu $val na konec v $list

zip($lists???) // Zkombinuje několik listů do jednoho multidimenzního seznamu
```

```
index($list, $value) // Vrátí pozici hodnoty v $list

list-separator(#list) // Vrátí separator seznamu
```

6.10.3 Funkce pro práci s Map

```
map-get($map, $key) // Vrátí hodnotu z $map asociovanou s $key

map-merge($map1, $map2)  // Sloučí dvě mapy do nové

map-remove($map, $keys???)    // Vrátí novou mapu bez daných $keys

map-keys($map) // Vrátí všechny použité klíče v $map

map-values($map)    // Vrátí seznam všech hodnot z mapy

map-has-key($map, $key) // Vrátí, zda mapa má hodnotu spojenou s daným klíčem

keywords($args)     // Vrátí klíčová slova předána do funkce
```

7 Stylus

Stylus je nejmladší z třetice testovaných CSS preprocesorů, jeho design ovlivnil jak SASS tak LESS. Stylus je stejně jako LESS psaný v javascriptu, veškerá instalace a použití probíhá pomocí node.js a jeho node package manageru. Stylus nabízí více možností pro zkušené uživatele, kteří si jej velmi často upravují pro vlastní potřeby, nabízí například vlastní Javascript API. Nepoužívá žádné znaky jako oddělovače resp. nejsou povinné, místo toho se zde využívá tabulátorů a mezer, stylus ovšem i syntaxi s klasickými složenými závorky, dvojtečkami a středníky umožní použít.

Stylus je vybaven všemi funkcemi, které SASS či LESS poskytují. Kromě toho poskytuje další pokročilé funkce a některé vylepšuje.

Jedná se opět o open-source projekt, všechny jeho zdrojové kódy najdete v repositářích GitHub (https://github.com/LearnBoost/stylus).

7.1 Syntaxe

Jak jsem již v úvodu zmínil, Stylus nevyžaduje ve své syntaxi žádné interpunkční znaménka. Více na ukázkovém příkladu zápisu.

Zápis v Stylus:

```
border-radius()
    -webkit-border-radius arguments
    -moz-border-radius arguments
    border-radius arguments
```

Vzhledem k tomu, že jsou tyto znaky nepovinné, lze je v pořádku využívat jako v klasickém CSS, nebo vynechat třeba jen složené závorky. Na příkladu využije dvojtečky a středníky.

Zápis v Stylus:

```
border-radius()
    -webkit-border-radius: arguments;
```

```
    -moz-border-radius: arguments;
    border-radius: arguments;
```

Není ovšem doporučeno míchat dvě různé syntaxe dohromady, může to pak později způsobovat nemilé problémy při kompilaci. Pokud potřebujete vložit kód v klasickém syntaxi CSS tj. ve složených závorkách s oddělovači, lze využít integrovanou funkci `@css`, do této funkce vložíme CSS style, v původní syntaxi a při kompilaci nenastane žádný problém.

Zápis v Stylus:

```
@css {
    body {
      font-size: 14px;
    }
 }
```

Výsledné CSS:

```
body {
  font-size: 14px;
}
```

7.2 Jak používat Stylus

7.2.1 Pomocí příkazové řádky

Stylus se nejčastěji kompiluje pomocí příkazové řádky, která umožní použití veškerých pokročilých funkcí tohoto preprocesoru.

```
// překlad souborů z adresáře /css.stylus do /css
stylus --watch css.stylus --out css
```

Opět můžete vidět příkaz `--watch`, tento příkaz nám stejně jako v ostatních preprocesorech zajistí sledování změn, které provedem ve zdrojovém souboru Stylusu a bude automaticky kompilovat výsledné CSS po uložení.

7.2.2 Pomocí aplikace třetích stran

I pro Stylus jsou vytvořené od vývojářů třetích stran aplikace pro snadnou kompilaci v grafickém rozhraní. Vzhledem k velikosti komunity a toho, že pokročilé funkce fungují zatím pouze při použití příkazové řádky, těchto aplikací není tolik a většinou jsou placené, já jsem pro svojí práci zvolil aplikaci CodeKit a pro pokročilé funkce příkazovou řádkou.

7.3 Proměnné

Proměnné se chovají stejně jako v LESS a SASS, jen mají odlišný zápis. Pro lepší čitelnost kódu můžete před název proměnné přidat klíčový znak `$`, který ovšem není povinný.

Zápis v Stylus:

```
font-size = 14px

font = font-size "Lucida Grande", Arial

body

  font font, sans-serif
```

Výsledné CSS:

```
body {

  font: 14px "Lucida Grande", Arial, sans-serif;

}
```

Stylus navíc obsahuje funkci, schopnou odkazovat na vlastnosti definované v dané třídě bez přiřazení jejich hodnoty do proměnné. Nejlepším příkladem je vertikální umístění prvku, pomocí logiky negativních okrajů. V příkladu využijeme i jednoduché matematické funkce.

Zápis v Stylus:

```
#logo

   position: absolute

   top: 50%
```

```
    left: 50%
    width: w = 150px
    height: h = 80px
    margin-left: -(w / 2)
    margin-top: -(h / 2)
```

Přímo v zápisu definice výšky a šířky, si určíme, že do proměnné `w` uložíme hodnotu `150px` a do `h` hodnotu `80px`. Lze využít i znaku `@` pro přístup k hodnotě obsahují danou vlastnost.

Zápis v Stylus:

```
#logo
    position: absolute
    top: 50%
    left: 50%
    width: 150px
    height: 80px
    margin-left: -(@width / 2)
    margin-top: -(@height / 2)
```

7.4 Interpolace

Stylus podporuju použití interpolace pomocí znaků `{}`, mezi tyto znaky pak přijde název volaného prvku. Jednoduchý příklad, kde si uložíme řetězec s klíčovými slovy border a –radius:

```
-webkit-{'border' + '-radius'}
```

Výsledné CSS:

```
-webkit-border-radius
```

Tyto interpolace se pak využívají k definování více selektorů najednou, v cyklech, podmínkách a dalších pokročilých funkcích.

Zápis v Stylus:

```
mojeSelektory = '#jeden,#dva,.tri

{mojeSelektory}

  background: #000
```

Výsledné CSS:

```
#jeden,

#dva,

.tri {

  background: #000;

}
```

7.5 Extend

Funkce přezvaná z preprocesoru SASS, slouží ke zkopírování všech vlastností daného elementu, tím, že se selektor ve kterém je extend použit, vloží jako selektor následující.

Zápis v Stylus:

```
.block {

  margin: 10px 5px;

  padding: 2px;

}


p {

  @extend .block;

  border: 1px solid #EEE;

}

ul, ol {
```

```
  @extend .block;
  color: #333;
  text-transform: uppercase;
}
```

Výsledné CSS:

```
.block, p, ul, ol {
  margin: 10px 5px;
  padding: 2px;
}
p {
  border: 1px solid #EEE;
}
ul, ol {
  color: #333;
  text-transform: uppercase;
}
```

7.6 Mixiny

Funkce známá již z LESS a SASS. V Stylus funguje stejně, zde mají navíc ještě stejnou syntaxi jako funkce, liší se v tom jak je co aplikováno v elementu. Rozdíl je vtom, že mixin má předem dán počet parametrů, která bude obsahovat, funkce může parametrů více a proto se nespecifikuje kolik. Více na příkladu níže, máme funkci `border-radius-fun()` a mixin `border-radius-mixin(n)`, který obsahuje právě jeden parametr `n`. Ve funkci jsme specifikovali, že volané argumenty se aplikují ve všech nastaveních. Chceme-li použít více parametrů v mixinech, oddělíme je čárkou.

Zápis v Stylus:

```
border-radius-fun()
  -webkit-border-radius arguments
  -moz-border-radius arguments
  border-radius arguments

border-radius-mixin(n)
  -webkit-border-radius n
  -moz-border-radius n
  border-radius n

form .success
  border-radius-fun 5px 2px 5px 6px
form .error
  border-radius-mixin 5px
```

Výsledné CSS:

```
form .succes {
  -webkit-border-radius: 5px 2px 5px 6px;
  -moz-border-radius: 5px 2px 5px 6px;
  border-radius: 5px 2px 5px 6px;
}
form .error {
  -webkit-border-radius: 5px;
  -moz-border-radius: 5px;
  border-radius: 5px;
}
```

7.6.1 Dědění v mixin

Mixin může využívat definice rodičovského elementu, který reprezentuje znak &, element pak působí na rodiče a ne jako další definice. Ideální příklad je pokud pro stylování tabulek, pro pruhované pozadí.

Zápis v Stylus:

```
pruhy(suda = #fff, licha = #eee)
   tr
     background-color licha
     &.even
     &:nth-child(even)
       background-color suda
 table
   pruhy()
   td
     padding 5px 10px
 table#uzivatele
   pruhy(#303030, #494848)
   td
     color white
```

Výsledné CSS:

```
table tr {
  background-color: #eee;
}
table tr.even,
table tr:nth-child(even) {
  background-color: #fff;
}
```

```
table td {
  padding: 5px 10px;
}
table#uzivatele tr {
  background-color: #494848;
}
table#uzivatele tr.even,
table#uzivatele tr:nth-child(even) {
  background-color: #303030;
}
table#uzivatele td {
  color: #fff;
}
```

Výsledek na webové stránce, dvě tabulky, každá jinak obarvená:

test 1	test 2	test 3
test 1	test 2	test 3
test 1	test 2	test 3

test 1	test 2	test 3
test 1	test 2	test 3
test 1	test 2	test 3

Obrázek 1 - Použití mixinu

7.6.2 Mixin v mixinu

Mixiny lze využívat jako další mixiny v návaznosti na své vlastní voliče a vlastností.

Zápis v CSS:

```
inline-list()
```

```
  li
    display inline
comma-list()
  inline-list()
  li
    &:after
      content ', '
    &:last-child:after
      content ''
ul
  comma-list()
```

Výsledné CSS:

```
ul li {
  display: inline;
}
ul li:after {
  content: ', ';
}
ul li:last-child:after {
  content: '';
}
```

7.7 Funkce

Funkce v Stylus mají zápis jako mixiny, ale můžete jim předat pokročilejší nastavení. Mohou provádět matematické funkce, práce s barvami apod. Na příkladu níže si

definujeme funkci `soucet(a, b)`, která obsahuje dva parametry, které se mezi sebou budou sčítat.

Zápis v Stylus:

```
add(a, b)
  a + b
```

Pokud tuto funkci zavoláme, sečte nám tyto dvě hodnoty a výsledek vloží před vlastnosti, do které jsem funkci přiřadili.

Zápis v Stylus:

```
body
   padding soucet(10px, 5)
```

Výsledné CSS:

```
body {
   padding: 15px;
 }
```

Na příkladu je také vidět, že Stylus nemá problém o jakou jednotku se jedná, to už uměl LESS. Pokud číslo 5 bude jako procento, přičte 5% z první hodnoty, jestliže bude jednotka nerozpoznaná, bere jí jako jednoduché číslo.

7.8 Pole

Stylus může vrátit v funkcích více hodnot než jen jednu, stejně můžete do jedné proměnné přiřadit více než jednu hodnotu, nenazývá se to přímo pole, ale je to této logice velmi podobné. V kódu níže, lze vidět uložené do proměnné `sizes` dvě hodnoty (15px a 10px), pokud se pak na dané hodnoty chceme odkázat, slouží zápis `sizes[0]`, 0 reprezentuje místo v poli. Jako ve většině programovacích jazycích, pole nezačíná od 1, ale od 0.

Zápis v Stylus:

```
sizes = 15px 10px
sizes[0]  // vrací 15px
```

Podobně se dají hodnoty zapsat i jako funkce.

```
sizes()
    15px 10px
sizes()[0]     // vrací 15px
```

Lze to celé zkombinovat s dalšími prvky a můžeme pak dostat například základní algoritmus bubble short.

```
sort(list, fn = null)
  // základní short funkce
  if fn == null
    fn = @(a, b) {
      a > b
    }
  // bubble sort
  for $i in 1..length(list) - 1
    for $j in 0..$i - 1
      if fn(list[$j], list[$i])
        $temp = list[$i]
        list[$i] = list[$j]
        list[$j] = $temp
  return list
sort('e' 'c' 'f' 'a' 'b' 'd') // vrátí 'a' 'b' 'c' 'd'
'e' 'f'
sort(5 3 6 1 2 4, @(a, b){
    a < b
  })
  // vrátí 6 5 4 3 2 1
```

Použití v praxi při tvorbě je při nejmenším minimální, slouží hlavně jako síla preprocesoru Stylus.

7.9 Vestavěné funkce

Seznam vestavěných funkcí, které lze kombinovat s ostatními funkcemi. Většina je stejná jako v LESS nebo SASS.

7.9.1 Práce s barvami

```
red(color[, value])      // vrátí hodnotu červené z rgb

green(color[, value])    // vrátí hodnotu zelené z rgb

blue(color[, value])     // vrátí hodnotu modré z rgb

alpha(color[, value])    // vrátí hodnotu průhlednosti z rgba

dark(color)              // zjistí jestli je color tmavá (true/false)

light(color)             // zjistí jestli je color světlá (true/false)

hue(color[, value])      // vrátí odstín dané barvy, nebo nastaví složku odstínů volitelnou hodnotu argumentu druhé

saturation(color[, value])   // vrátí sytost dané barvy, nebo nastaví složku sytosti na volitelnou hodnotu argumentu druhé

lightness(color[, value])    // vrátí světlost dané barvy, nebo nastaví složku světlosti k volitelnému hodnoty argumentu druhé

hsla(color | h,s,l,a)    // vratí HSLA z rgba nebo jednotlivých kanálů hsla

hsl(color | h,s,l)       // vrátí HSL z rgb nebo jednotlivých kanálů hsl
```

```
rgba(color | r,g,b,a)    // vrátí HEX z jednotlivých kanálů r,g,b,a

rgb(color | r,g,b)       // vrátí hex hodnotu z r,g,b kanálů

lighten(color, amount)   // zesvětlí barvu o procenta (pokud jsou zadaná) nebo o zadaný počet

darken(color, amount)    // ztmaví barvu o procenta (pokud jsou zadaná) nebo o zadaný počet

desaturate(color, amount)

saturate(color, amount)

complement(color)

invert(color)

grayscale(color)

tint(color, amount)

shade(color, amount)

luminosity(color)

contrast(top[, bottom])

transparentify(top[, bottom, alpha])
```

7.9.2 Matematické funkce

```
abs(unit)      // absolutní hodnota

ceil(unit)     // zaokrouhlení nahoru

floor(unit)    // zaokrouhlení dolů

round(unit)    // logické zaokrouhlení 5.5 = 6, 5.4 = 5

sin(angle)     // vrací sinus

cos(angle)     // vrací cosinus

tan(angle)     // vrací tangent

min(a, b)      // vrací minimální hodnotu
```

```
max(a, b)      // vrací maximální hodnotu
even(unit)     // true/false je-li jednotka jednotka
odd(unit)      // true/false je-li jednotka délka
sum(nums)      // vrací sumu
avg(nums)      // vrací průměr
```

7.9.3 Další funkce

```
push(expr, args…)        // vloží mezi args, hodnotu expr např. čárku
unshift(expr, args…)     // vloží hodnotu expr za hodnoty args
keys(pairs)              // vrátí zadané klíče hodnot
pairs = (one 1) (two 2) (three 3)
 keys(pairs)   // vrátí one two three
values(pairs)            // obdobné jako předchozí funkce, zde se vrátí hodnoty
list-separator(list)     // vrátí separátor který odděluje hodnoty
typeof(node)             // vrátí typ hodnoty (unit, rgba, ...)
unit(unit[, type])       // vrátí typ jednotky nebo typ dané jednotce přidá
base-convert(num, base, width)    // převede číslo do zadané číselné soustavy a o dané šířce (defaultní je dvojková soustava)
```

7.10 Funkce url()

Funkce url() je dostupná pouze v preprocesoru Stylus, jedná se o velmi užitečnou funkci pokud chcete ušetřit každý http request při načítání webu, což se na velkém

portále, který obsahuje velmi mnoho různých obrázků, stylů a javascript souborů, velmi hodí.

Funkce přepisuje relativní odkazy ve vašem Stylus souboru, nejčastěji u vlastnosti background, adresu přepíše do formátu base64 v Data URI schéma. [7]

Výsledné bez použití url() CSS:

```
ul li.complete {
    padding-left: 20px;
    background: white url('image/pozadi.png') no-repeat scroll left top;
}
```

Výsledné s použitím url() CSS:

```
ul li.complete {
     padding-left: 20px;
     background: white
     url('
     BAAAAAQAQMAAAAlPW0iAAAABlBMVEUAAAD///+l2Z/dAAAAM0lEQ
     VR4nGP4/5/h/1+G/58ZDrAz3D/McH8yw83NDDeNGe4Ug9C9zwz3g
     VLMDA/A6P9/AFGGFyjOXZtQAAAAAElFTkSuQmCC') no-repeat
     scroll left top;
}
```

8 Hodnocení zkoumaných preprocesorů

Preprocesor LESS mi sedl nejvíce, jeho syntaxe je naprosto stejná jako v klasickém CSS, proto když jsem s preprocesory před rokem a půl začínal byl přechod velmi snadný. Mohl jsem využít CSS styly, které jsem měl již z předchozích projektů. Seznamoval jsem se s funkcemi a postupně je zaváděl do praxe. LESS nemá doladěné pokročilé funkce jako jsou cykly, podmínky, ale v praxi těchto funkcí využijete jen velmi málo. Proto bych se při výběru vůbec nerozhodoval podle seznamu funkcí. V LESS je psaná drtivá většina frameworků, např. Bootstrap.

SASS u mě skončil na pomyslném druhém místě. SASS používá dvě různé syntaxe, jedná se nazývá SASS, nepoužívá středníky ani složené závorky a místo toho odsazuje mezerami či tabulátory. Druhou syntaxí je SCSS, která vypadá jako klasické CSS a měly by být i plně kompatibilní (podle oficiální dokumentace), syntaxe je pak úplně stejná jako LESS.

U Stylusu není povinné vynechat speciální znaky, u SASS ano! Nejspíš proto vytvořili druhou syntaxi, která se více blíží k CSS. Pokud jsem chtěl pak využít některé frameworky byl třeba napsaný jen v první syntaxi (bez speciálních znaků) a stou si SCSS neporadil.

Na Stylusu je vidět, že je nejmladší, a že se jeho autor poučil z chyb ostatních. Zároveň implementoval neskutečné množství drobných vychytávek, někdo si to chválí někdo zas ne. Seznam těch nejpoužívanějších:

- Mixiny bez zbytečných teček na začátku
- Podmíněné styly
- Mixiny vnořené v dalších mixinech
- Knihovna NIB, všechny užitečné mixiny na jednom místě
- Inlining obrázků do BASE64 pomocí funkce url()
- Někdo také ocení vynechání označení proměnných speciálními znaky

Pokud jste více programátor, a s CSS si moc netyká volte SASS (SCSS). Pokud vám CSS nedělá problém a rádi si vytváříte vlastní funkce nebo využíváte více programátorských funkcí volte Stylus.

8.1 Tabulka srovnání zkoumaných preprocesorů

Moje osobní hodnocení zanesené do tabulek. Hodnotil jsem syntaxi, možnosti a jednoduchost kompilace a chybové hlášení, na stupnici od 1 do 3 (kde 3 je nejlepší). Dále jsem srovnával funkce, které jsou v preprocesorech rozdílné.

	Syntaxe	**Kompilace**	**Chybové hlášky**	**@import**	**Podmínky**
LESS	3/3	3/3	3/3	Ano	when
SASS	2/3	2/3	2/3	Ano	@if, @else if, @else, then
Stylus	1/3	1/3	1/3	Ano + @require	If, else if, else, unless

Tabulka č. 3 - Srovnání zkoumaných preprocesorů část 1

	Cykly	**Objekty**	**url()**	**Vlastní funkce**	**Zanořování mixinu**
LESS	for	Ne	Ne	Ne	Ne
SASS	for, while	Ne	Ne	Ne	Ano
Stylus	for	Ano	Ano	Ano	Ano

Tabulka č. 4 - Srovnání zkoumaných preprocesorů část 2

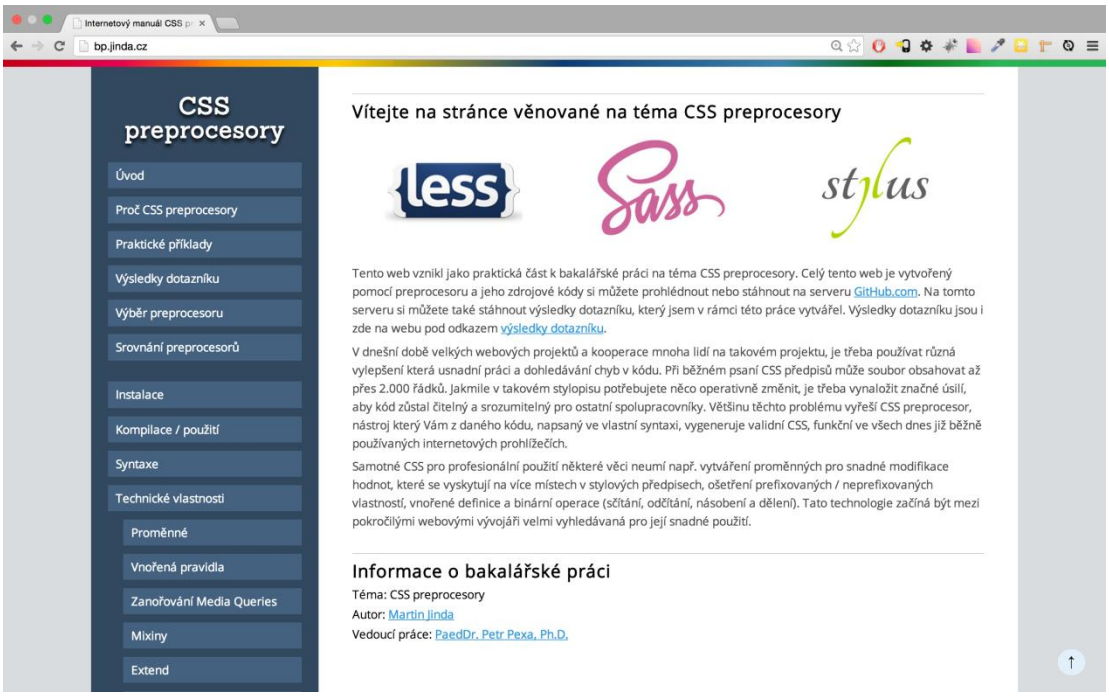
9 Praktická část

Praktickou částí mojí bakalářské práce bylo vytvořit webovou stránku pomocí pro mě nejvhodnějšího preprocesoru. Web jsem tedy vytvořil pomocí preprocesoru LESS, základ tvoří jednoduchý PHP systém. Zdrojové soubory webu jsou dostupné online, v git repositáři, na serveru Github.com, na adrese http://www.github.com/martinjinda/bp_web. Web je validní dle W3C, pomocí Apache funkce, mod_rewrite, má web „SEO friendly“ tvary adres, pro lepší zapamatování a orientaci uživatelů. Web je také plně responsivní, lze jej tedy prohlížet na jakémkoliv zařízením.

Stylopis webu je kompilován z jednoho hlavního LESS souboru do kterého je pomocí funkce @import vloženo 10, parciálních, LESS souborů. Tento způsob nezpomaluje tolik načítání webu, například na mobilech, kde je využíváno připojení k internetu mobilní operátor.

Na webu najdete praktické příklady použití preprocesorů v praxi, na kterých demonstruji jejich výhody oproti klasickému zápisu CSS souboru. Vložil jsem tam také veškeré zjištěné informace, které jsem během práce nashromáždil, tyto informace poslouží nejvíce začínajícím tvůrcům stránek, kteří chtějí zavést preprocesory do jejich pracovního postupu.

Vzhledem ke kladnému ohlasu uživatelů, plánuji do budoucna přesunutí celého webu na doménu, která bude více odrážet účel tohoto webu. Aktuální adresa je http://bp.jinda.cz.



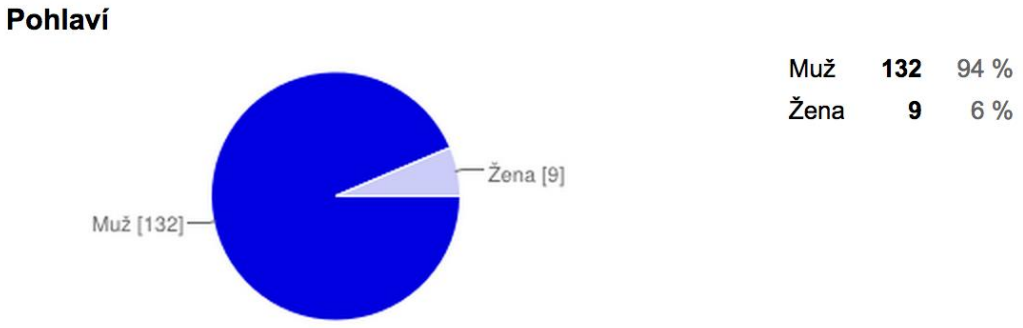
Obrázek 2 - Praktická část - http://bp.jinda.cz

10 Dotazník

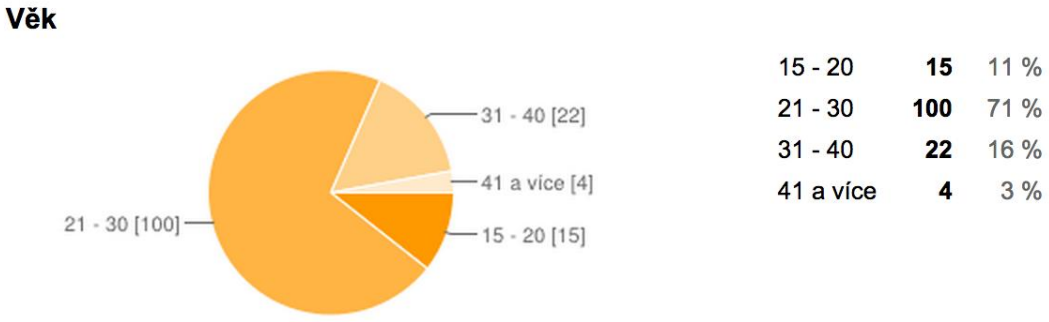
V rámci práce jsem vytvořil dotazník o rozšířenosti preprocesorů v odborné veřejnosti. Nejvíce mě zajímalo, zda preprocesory vůbec v praxi využívají, jaký preprocesor nejčastěji používají, zda jim pomáhá při tvorbě webových stránek, kde se s touto technologií setkali a zda využívají také další nástroje, které dokáží velmi zjednodušit a zorganizovat vývoj projektu při kooperaci více lidí.

Dotazník byl vytvořený pomocí nástroje Google Form. Dotazník jsem zaslal do 5 firem zabývající se tvorbou webových stránek nebo webového rozhraní a do skupin na sociální síti Facebook, kde se probírají nové technologie v oblasti tvorby webových stránek. Celkový počet respondentů tak byl 500. Počet odpovědí bylo 142.

10.1 Souhrn odpovědí

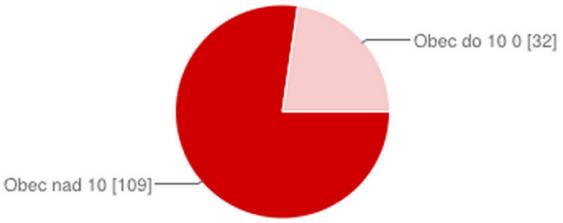


Obrázek 3 - Dotazník - Pohlaví



Obrázek 4 – Dotazník - Věk

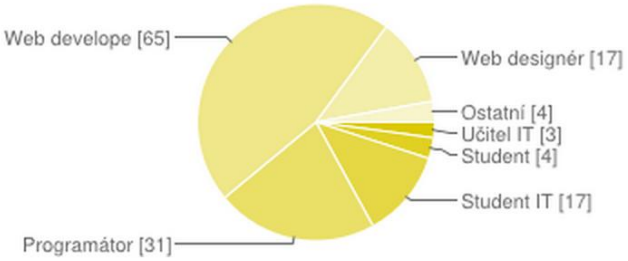
Bydliště



Obec nad 10 000 obyvatel	**109**	77 %
Obec do 10 000 obyvatel	**32**	23 %

Obrázek 5 - Dotazník - Bydliště

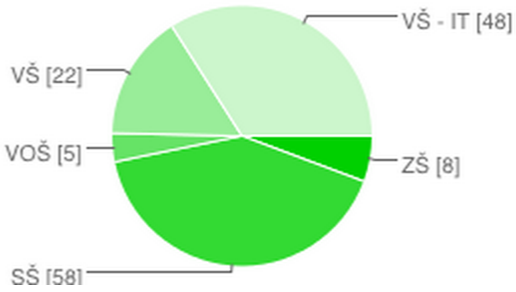
Povolání



Učitel IT	**3**	2 %
Student	**4**	3 %
Student IT	**17**	12 %
Programátor	**31**	22 %
Web developer	**65**	46 %
Web designér	**17**	12 %
Ostatní	**4**	3 %

Obrázek 6 – Dotazník - Povolání

Vzdělání



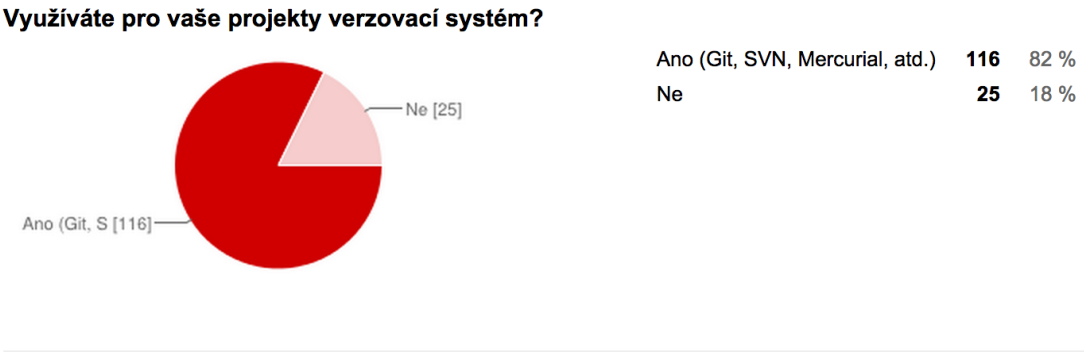
ZŠ	**8**	6 %
SŠ	**58**	41 %
VOŠ	**5**	4 %
VŠ	**22**	16 %
VŠ - IT	**48**	34 %

Obrázek 7 - Dotazník - Vzdělání

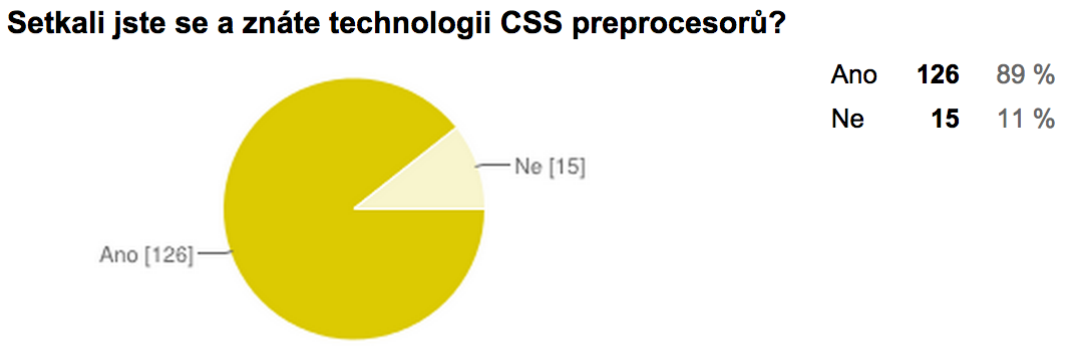
Obrázek 8 - Dotazník - Vytváříte webové prezentace na internetu?

Obrázek 9 - Dotazník - Jaké technologie při tvorbě webů používáte?

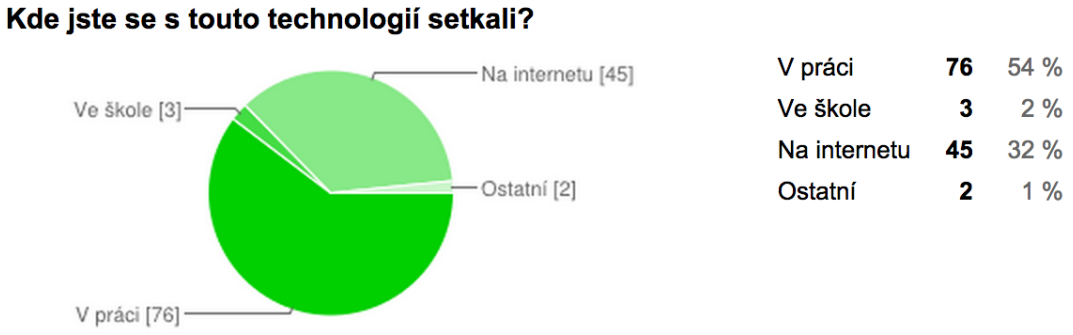
Obrázek 10 – Dotazník - Zajímáte se o nové techniky v tvorbě webových stránek?



Obrázek 11 – Dotazník -Využíváte pro vaše projekty verzovací systém?

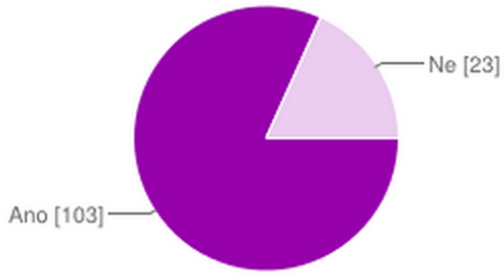


Obrázek 12 - Dotazník - Setkali jste se a znáte technologii CSS preprocesorů?



Obrázek 13 - Dotazník - Kde jste se s touto technologií setkali?

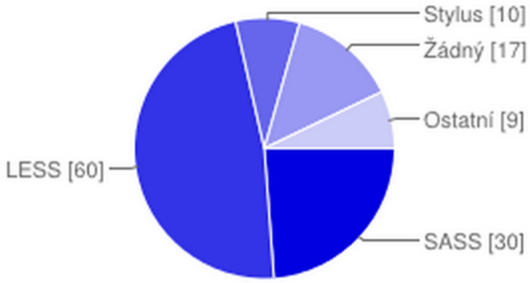
Používáte CSS preprocesory aktivně?



Ano	**103**	73 %
Ne	**23**	16 %

Obrázek 14 - Dotazník - Používáte CSS preprocesory aktivně?

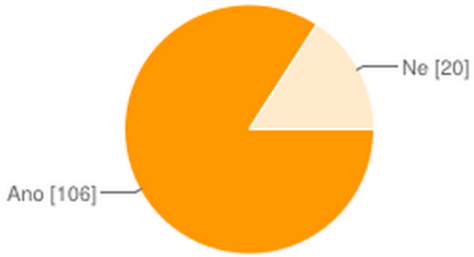
Jaký CSS preprocesor používáte?



SASS	**30**	21 %
LESS	**60**	43 %
Stylus	**10**	7 %
Žádný	**17**	12 %
Ostatní	**9**	6 %

Obrázek 15 - Dotazník - Jaký CSS preprocesor používáte

Pomáhají vám CSS preprocesory k urychlení práce na webových projektech?



Ano	**106**	75 %
Ne	**20**	14 %

Obrázek 16 – Dotazník - Pomáhají vám CSS preprocesory k urychlení práce na webových projektech?

Obrázek 17 - Dotazník - Uvítali byste možnost obsáhlého internetového manuálu na téma CSS preprocesory?

11 Závěr

Během psaní bakalářské práce jsem prostudoval veškeré dostupné informace o technologii CSS preprocesorů, která začíná být velmi žádaná v praxi, mezi IT firmami zabývající se tvorbou internetových stránek, webových portálů a uživatelského rozhraní na internetu a mezi IT odborníky z téhož oboru. Mimo samotných preprocesorů jsem se seznámil i s dalšími nástroji, které dokáží také velmi efektivně vylepšit pracovní postupy např. Grunt, Bower a další. Z dostupných CSS preprocesorů jsem podle určitých kritérií vybral tři, které jsem dále zkoumal a zkoušel na praktických příkladech, mezi tyto tři preprocesory patří LESS, SASS a Stylus. Vytvořil jsem webovou stránku, kde jsem uvedl veškeré možnosti zkoumaných preprocesorů, rozebral jejich funkce, návod na instalaci, jak je používat, jak je začlenit do pracovního procesu a jaké výhody přinesou oproti klasickému CSS. Na této webové stránce jsou také ukázky kódů praktických příkladů, na kterých sem demonstroval sílu preprocesoru oproti klasickému CSS.

Mnoho webových vývojářů se bojí přejít ze svých osobních pracovních postupů při tvorbě webových stránek. Často slyším názory, že preprocesory jsou velmi složité a že akorát prodlužují práci. S tímto tvrzením naprosto nesouhlasím, jediný čas navíc, který se musí preprocesorům věnovat je při instalaci a nastudování syntaxe daného preprocesoru. Proto jsem jako praktickou část vytvořil web, kde jsem veškeré informace a postupy uvedl, abych začátečníkům v preprocesorech usnadnil start.

Potvrdil se můj předpoklad, že pro začátek je dobré zvolit preprocesor LESS, který má totožnou syntaxi jako klasické CSS, přechod na preprocesory tedy není tak razantní. SASS a Stylus mají „programátorskou“ povahu, obsahují velmi pokročilé funkce (cykly, podmínky, definice vlastních funkcí a další), velmi váhám nad použitelností v praxi.

Jako praktickou část práce jsem vytvořil webovou stránku, na které jsem uvedl veškeré mnou zjištěné informace, rady a návody, jak s preprocesory rychle začít a jak je používat. Celá webová stránka je vytvořená pomocí preprocesoru LESS. Na web jsem umístil také různé praktické příklady, vytvořené pomocí všech tří zkoumaných preprocesorů, na kterých demonstruji, že preprocesory dokáží vylepšit pracovní postupy při tvorbě webových stránek. Web je umístěn na adrese http://bp.jinda.cz.

Seznam použité literatury a zdrojů

[1] KOSEK, Jiří. Historie a vývoj HTML. In: *HTML guru* [online]. 2013 [cit. 2014-11-27]. Dostupné z: http://htmlguru.cz/uvod-historie.html

[2] Co je DHTML. In: *Adaptic, s.r.o.* [online]. 2013 [cit. 2014-11-27]. Dostupné z: http://www.adaptic.cz/znalosti/slovnicek/dhtml/

[3] 8 CSS preprocessors to speed up development time. *8 CSS preprocessors to speed up development time* [online]. 2010 [cit. 2014-12-08]. Dostupné z: http://www.catswhocode.com/blog/8-css-preprocessors-to-speed-up-development-time

[4] *LESS: Dynamický jazyk pro tvorbu stylesheetů* [online]. 2012 [cit. 2014-11-27]. Dostupné z: http://www.lesscss.cz/

[5] *LESS: Dynamický jazyk pro tvorbu stylesheetů* [online]. 2012 [cit. 2014-11-27]. Dostupné z: http://www.lesscss.cz/

[6] SASS - Čiastočné súbory a import (3). *SASS - Čiastočné súbory a import (3)* [online]. 2014, č. 3 [cit. 2014-12-02]. Dostupné z: http://www.zajtra.sk/programovanie/1154/sass-ciastocne-subory-a-import-3

[7] *Url() Stylus* [online]. 2013 [cit. 2014-12-08]. Dostupné z: http://learnboost.github.io/stylus/docs/functions.url.html

[8] SASS. *Sass: Syntactically Awesome Style Sheets* [online]. 2006 [cit. 2014-06-03]. Dostupné z: http://www.sass-lang.com/ LESS. Less [online]. 2009 [cit. 2014-06-03]. Dostupné z: http://www.lesscss.org/

[9] STYLUS. *Stylus - Expressive, dynamic, robust CSS* [online]. 2009 [cit. 2014-06-03]. Dostupné z: http://learnboost.github.io/stylus/

[10] GOOGLE. *Working with CSS Preprocessors* [online]. 2014 [cit. 2014-06-03]. Dostupné z: https://developer.chrome.com/devtools/docs/css-preprocessors

[11] *Sass And LESS: An Introduction To CSS Preprocessors* [online]. 2012 [cit. 2014-12-09]. Dostupné z: http://www.vanseodesign.com/css/css-preprocessors/

[12] MICHÁLEK, Martin. *Průvodce CSS preprocesory* [online]. 2014 [cit. 2014-06-03]. Dostupné z: http://www.vzhurudolu.cz/blog/12-css-preprocesory-1

[13] Internet Explorer's CSS rules limits. In: *Internet Explorer's CSS rules limits* [online]. 2012 [cit. 2014-12-09]. Dostupné z: http://stackoverflow.com/questions/9906794/internet-explorers-css-rules-limits

[14] VERRECCHIA, Jonathan. *Deep Dive Into CSS Preprocessors* [online]. 2012 [cit. 2014-12-09]. Dostupné z: http://www.slideshare.net/verekia/deep-dive-into-css-preprocessors?qid=868d8ae0-a4d5-4001-a967-a7c5e61ec8e0&v=qf1&b=&from_search=1

[15] PAGE, Luke. LESS VS SASS VS STYLUS. In: *LESS VS SASS VS STYLUS* [online]. 2013 [cit. 2014-12-09]. Dostupné z: http://www.scottlogic.com/blog/2013/03/08/less-vs-sass-vs-stylus.html

[16] 10 LESS CSS Examples You Should Steal for Your Projects. JOHNSON, Joshua. *10 LESS CSS Examples You Should Steal for Your Projects* [online]. 2011 [cit. 2014-12-09]. Dostupné z: http://designshack.net/articles/css/10-less-css-examples-you-should-steal-for-your-projects/

Seznam obrázků

Seznam tabulek