Univerzita Karlova

Pedagogická fakulta

Katedra informačních technologií a technické výchovy

BAKALÁŘSKÁ PRÁCE

# Vývoj a trendy stylování www stránek

# Development and trends websites styling

Radim Šmarda

| | |
|---|---|
| Vedoucí práce: | PhDr. Josef Procházka, Ph.D. |
| Studijní program: | Specializace v pedagogice |
| Studijní obor: | Informační technologie se zaměřením na vzdělávání |

2018

Odevzdáním této bakalářské práce na téma Vývoj a trendy stylování www stránek potvrzuji, že jsem ji vypracoval pod vedením vedoucího práce samostatně za použití v práci uvedených pramenů a literatury. Dále potvrzuji, že tato práce nebyla využita k získání jiného nebo stejného titulu.

Praha 20. 4. 2018

Rád bych touto cestou vyjádřil poděkování panu PhDr. Josefu Procházkovi, Ph.D. za jeho cenné rady a trpělivost při vedení mé bakalářské práce.

**ABSTRAKT**

Tato bakalářská práce se zabývá aktuálními trendy ve stylování www stránek a podává přehled o historických milnících vývoje kaskádových stylů, o jejich současných vlastnostech a o nejběžnějších CSS frameworcích a preprocesorech. Práce také obsahuje interaktivní ukázku, na které jsou aktuální CSS vlastnosti představeny ve formě výukové prezentace. Ty jsou podrobeny testu v současných verzích prohlížečů a výsledky jsou rovněž součástí této práce.

**KLÍČOVÁ SLOVA**

CSS, kaskádové styly, web, vývoj

**ABSTRACT**

This bachelor thesis speaks about current trends in styling of webpages and shows the summary of important historical points in the development of cascading style sheets and their actual properties. It also deals with the topic of the most common CSS frameworks and preprocessors. This bachelor thesis also contains an interactive illustration, which shows the CSS properties in an educational presentation. The illustrated properties have then been tested in present – day browsers and the test results are also presented in this bachelor thesis.

**KEYWORDS**

CSS, Cascading Style Sheets, web pages, development

# Obsah

# Úvod

V dnešní době, kdy se přesouvá velké množství aplikací a dat do cloudu a preferují se mobilní zařízení, získávají webové aplikace na stále větší popularitě. Je proto nutné vymýšlet a držet krok s moderními technologiemi a trendy. Jak již bývá zvykem, vývoj jazyků ať už HTML, CSS, nebo jiných je pomalejší, než bychom si přáli, a proto vzniká nepřeberné množství frameworků a preprocesorů, které se snaží řešit nedostatky současných technologií a usnadňují tak práci vývojářům.

Cílem této práce je podat přehled historického vývoje kaskádových stylů, současných trendů, možností ve stylování www stránek a seznámit se s nejběžnějšími CSS frameworky a preprocesory. V praktické části je cílem vytvořit interaktivní webovou prezentaci s přehledem aktuálních CSS vlastností pro výukové účely na základních, středních, ale i vysokých školách. Tuto prezentaci, potažmo CSS vlastnosti, následně podrobit testu funkčnosti v aktuálních verzích webových prohlížečů.

Přínosem této práce je především vznik publikace, která popisuje aktuální trendy ve stylování webových stránek, a vytvoření prezentace, která bude sloužit jako výukový nástroj ve školách, nebo u začínajících webdesignerů a vývojářů.

# 1 Kaskádové styly

Kaskádové styly neboli zkráceně CSS z anglického Cascading Style Sheets je jazyk popisující zobrazení dokumentů psaných ve značkovacích jazycích HTML a XML. Kaskádové jsou proto, že je to jejich charakteristická vlastnost a při jejich psaní se využívá tzv. kaskády. To znamená, že na sebe mohou vrstvit různé definice stylů.

Spolu se značkovacím jazykem HTML tvoří hlavní dvojici při tvorbě webových stránek. Ačkoliv kaskádové styly nebyly historicky jedinou technologií pro stylování dokumentů, dnes jen stěží najdeme webovou stránku, která by kaskádové styly neobsahovala. Hlavním účelem kaskádových stylů je formátovat všechny prvky dokumentu jako je změna barvy, odsazení, okraje, animace atd. To, co je možné pomocí CSS nastylovat, je dáno specifikací, kterou vyvíjí konscorcium W3C a především podporou ve webových prohlížečích.

Jsou-li kaskádové styly správně používány, umožňují oddělení vzhledu dokumentu od jeho struktury, a tím je možné dosáhnout větší přehlednosti kódu, jednodušší změny vzhledu a zvýšení přístupnosti stránky. Kaskádové styly je možné v dokumentu zapisovat třemi způsoby. Jako inline styl, tzn. přímý zápis stylu pomocí atributu style danému elementu, jako zápis stylů do elementu <style> celému dokumentu, nebo jako připojení externího souboru pomocí elementu <link>. Jestliže je využito poslední varianty, je možné využít dalších pozitivních vlastností kaskádových stylů, jako je rychlejší načítání stránky s využitím cache paměti v prohlížeči, možnost minifikace a spojování CSS souborů do jednoho.

## 1.1 Historický vývoj kaskádových stylů

Než vznikly kaskádové styly, webové stránky se od dnešní podoby značně lišily. V roce 1990, kdy vzniklo HTML, byla jediným cílem prezentace obsahu v čisté strukturované formě s minimem HTML značek (Cyroň, 2006, s. 16). Mělo být pouze zřejmé, co je nadpis, co je odstavec apod. První prohlížeč na světě, který zobrazoval webové stránky, byl NeXT. Tento prohlížeč formátoval dokument na základě jednoduchého stylového předpisu.

V roce 1992 započal vývoj webového prohlížeče NCSA Mosaic, který byl v roce 1993 vydán ve verzi 0.1 (Bos, 2016). Díky tomu, že byl bezplatný, stal se web ještě více populární. Na druhou stranu Mosaic upustil od konceptu stylování a kromě omezených možností, jako byla změna barvy a písma, nebylo možné dokument jakkoliv stylovat. To se nelíbilo

mnohým autorům webových stránek. A protože jeden z hlavních programátorů Mosaicu Marc Andreessen, chtěl vyjít autorům www stránek vstříc, stal se spoluzakladatelem firmy Netscape, která si dala za cíl vytvořit nový prohlížeč s podporou stylování.

V roce 1994, tři dny předtím, než společnost Netscape oznámila dostupnost svého prohlížeče, vydal Håkon Wium Lie (člen World Wide Web Consortium, zkráceně W3C) první návrh kaskádního HTML stylu (CHSS). Prohlížeč Netscape tedy neměl podporu stylů, ale místo toho byly zabudovány vlastnosti pro řízení vzhledu přímo do značek HTML (Cyroň, 2006, s. 16). To zahájilo éru nekompatibility mezi prohlížeči a původně čistý strukturovaný HTML dokument obsahoval zbytečný kód navíc.

Jeden z prvních lidí, který reagoval na první návrh kaskádových stylů, byl Bert Bos. V té době vyvíjel prohlížeč Argo s podporou Stream-based Style Sheet Proposal (SSP) stylů, které by mohly být použity i mimo HTML. Tento koncept univerzálního jazyka pro jakékoli značkovací jazyky se zdál jako optimální, a proto jej převzal i Håkon Wium Lie pro svůj návrh CHSS. Jelikož CHSS styl měl být také univerzální, bylo z jeho názvu odstraněno písmeno H značící HTML. Tím vzniklo dodnes užívané označení CSS. Kromě návrhu CHSS a SSP vzniklo dalších asi 10 návrhů stylů, jako např. DSSSL a SGML, které se ovšem neujaly (Bos, 2016).

První prohlížeč, který podporoval CSS, byl v roce 1996 Internet Explorer 3 od Microsoftu. Ovšem v té době kaskádové styly ještě nebyly nikým standardizovány, takže implementace v tomto prohlížeči se s pozdějšími předpisy příliš neshoduje (Cyroň, 2006, s. 17). Podpora CSS byla navíc omezena pouze na změnu písma a barvy.

### 1.1.1 CSS 1

První standardizovaná verze CSS 1 od konsorcia W3C byla vydána na konci roku 1996, tedy čtyři měsíce po vydání Internet Exploreru 3. Konkurence v podobě Netscape se k CSS stavěla skepticky a netoužila po dodržování standardů. Netscape v té době slavil úspěch se svým jazykem JavaScript, kterým bylo možné formátovat HTML dokument (Bos, 2016). Proto vznikla technologie JavaScript Style Sheets (JSS), která využívala JavaScript pro práci se styly. Tato technologie se nikdy neujala a kromě Netscape ve verzi 4 se jinde neobjevila.

Solidní podpory se kaskádové styly verze 1 dočkaly až v roce 1997 v Internet Exploreru 4, který podporoval všechny vlastnosti specifikace CSS 1. Tato verze kaskádových stylů v sobě zahrnuje např. vlastnosti písma, zarovnání textu, tabulky, mezery slov, okraje atd. Protože weboví designeři potřebovali metodu pro přesné umístění elementu na webové stránce a současné CSS 1 to neumělo, vznikl ještě standard CSS-P (CSS-Pozicování). Tento standard byl jak v Internet Exploreru, tak v Netscape 4.0 zahrnut v rámci podpory CSS 1 a přidával podporu k pozicování elementů v dokumentu. U Netscape podpora CSS 1 nebyla nikterak velká a mnoho vlastností nebylo implementováno vůbec. Internet Explorer kromě podpory vlastností z CSS 1 přidává navíc i filtry a užitečnou pseudotřídu a:hover (Janovský). Třetím prohlížečem, který v té době podporoval většinu CSS 1 vlastností, byla norská Opera ve verzi 3.5 z roku 1998 (Schengili-Roberts, 2014).

### 1.1.2 CSS 2

V květnu 1998 bylo vydáno CSS 2, které kromě podpory všech vlastností z CSS 1 i CSS-P přidává několik vlastností nových. (Teague, 2005, s. 9). Velký přínos je specifikace pro různá média, tzn. že je možné např. nastavit styl pro dokument, jiný styl pro tisk a jiný styl pro mobilní telefon. Kromě toho byly doplněny některé drobné funkce jako např. stíny, formátování tabulek, vlastní kurzor a jiné.

V roce 2006 byla vydána revize CSS 2.1, která doplňuje některé funkce a primárně opravuje všechny publikované chyby CSS 2. Tato revize probíhala do června 2011 a poté se vývoj zaměřil na novou specifikaci kaskádových stylů. Tedy teprve v roce 2011 dalo konsorcium W3C oficiální doporučení pro implementaci v prohlížečích. Podpora CSS 2 byla obzvláště v Internet Exploreru mnohdy zvláštní, a proto je celkem překvapivou skutečností, že Microsoft, který v roce 2009 vydal svůj Internet Explorer 8, prohlašoval, že je to první prohlížeč s úplnou podporou specifikace CSS 2.1 (Gasston, 2016, s. 23-24).

### 1.1.3 CSS 3

Práce na specifikaci CSS 3 začaly v roce 1998, tedy už v době prvního vydání CSS 2. V té době byla implementace CSS 2 v prohlížečích značně nekonzistentní, a proto se vývoj nové verze pozastavil a raději se konscorcium W3C věnovalo vývoji CSS 2.1. Až v roce 2005 se vrátilo ke specifikaci nového standardu a započal proces úprav a hodnocení návrhů.

Po dlouhou dobu byl nejrozšířenějším prohlížečem Internet Explorer, který ovšem nevykazoval známky, že by měl někdy podporovat CSS 3 (Gasston, 2016, s. 24). Proto se rozjel konkurenční boj u ostatních prohlížečů, které chtěly přinést vývojářům nejnovější technologie. Jelikož postupem času byla specifikace CSS 3 téměř hotová, bylo zavádění nových funkcí poměrně snadné.

Vývoj hlavních vlastností CSS 3 je dnes již dokončen, nebo téměř před dokončením a žádný z hlavních webových prohlížečů nemá s novou verzí vážnější problémy. Ovšem jazyk CSS 3 není tak úplně nová verze jako předchozí CSS 2 a CSS 1, ale stal se modulárním jazykem. Protože konscorcium W3C vědělo, že vývoj ať už specifikace, nebo implementace v prohlížečích může trvat mnoho roků, rozdělilo funkce na jednotlivé moduly. Z tohoto důvodu neexistuje jeden obrovský dokument specifikace CSS 3 jako v případě CSS 2.1, ale máme kratší specifikace např. CSS 3 Basic User Interface, Media Queris, Selectors Level 3 atd. Ty mohou prohlížeče implementovat samostatně (Gaston, 2015, s. 30). Některé moduly jsou revizí CSS 2.1, některé jsou zcela nové, a proto jsou opatřeny číselnou úrovní, značící kolika iteracemi prošly.

Každý z těchto modulů je v nějakém stavu, tento stav označuje, v jaké fázi procesu doporučení se daný modul nachází (Gaston, 2016, s. 25). Když konscorcium W3C přijme navržený dokument jako součást specifikace CSS, přidělí mu stav Working Draft (pracovní návrh). To znamená, že je dokument publikován a čeká na hodnocení. Hodnocení provádí většinou pracovní skupiny W3C a vývojáři prohlížečů. Dokument může v tomto stavu zůstat dlouhou dobu a může se stát, že již v této fázi zanikne.

Pokud se nějaký výrobce prohlížeče rozhodne v této fázi implementovat danou funkci ve svém produktu, využívají se tzv. vendor prefixes. To jsou jednoduché prefixy určité CSS vlastnosti, které označují, že se jedná o funkci, která je pouze testovací a může se změnit. Každý z výrobců používá vlastní prefixy a ty jsou následující:

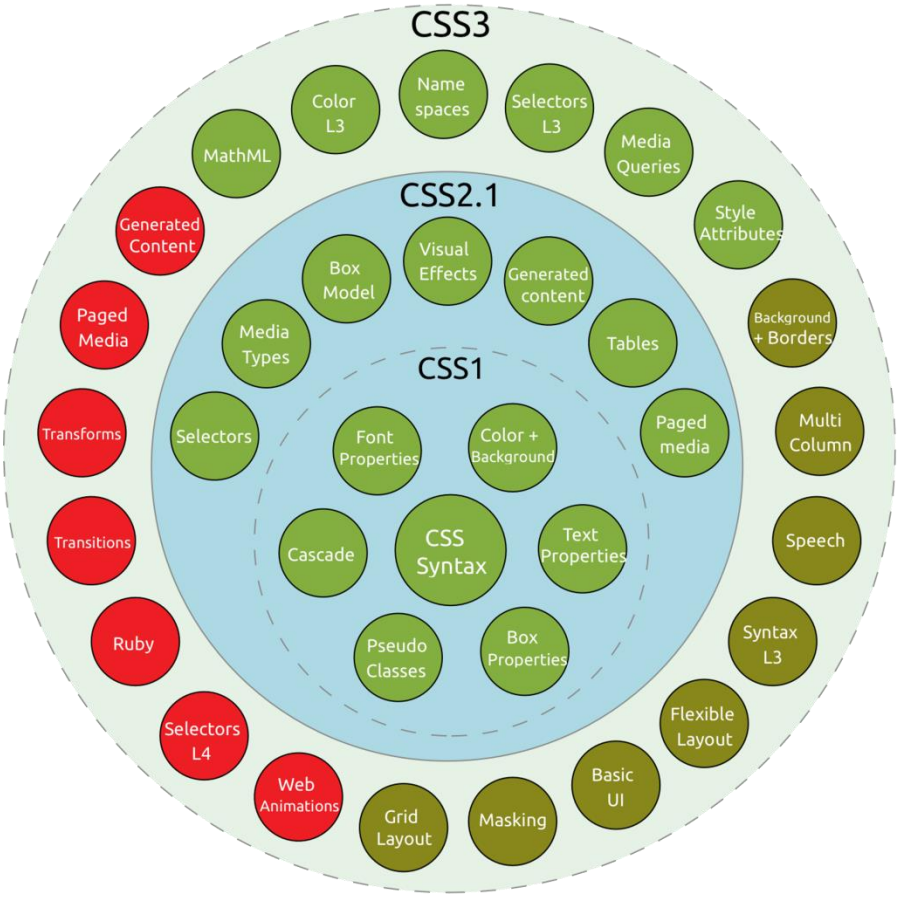
- -moz- prefix pro prohlížeč Firefox, potažmo celé jádro Gecko,
- -webkit- prefix pro prohlížeče postavené na jádře webkit, např. Safari, Chrome, nová Opera,
- -ms- prefix pro Internet Explorer a Microsoft Edge,

- -o- prefix pro původní verzi prohlížeče Opera, která ještě nebyla postavená na jádře WebKitu potažmo Blinku.

Jakmile projde dokument specifikace touto fází, změní se jeho stav na Last Call. To znamená, že hodnocení končí a dokument postupuje do další fáze.

Další úrovní je Candidate Recommendation. V tomto stavu mohou výrobci prohlížečů začít implementovat nové vlastnosti a získat tak odezvu z reálného používání. Tentokrát již bez CSS prefixů. Jakmile daný modul implementují nejméně dva prohlížeče a neobjeví se žádné technické problémy, dokument přechází do stavu Proposed. To znamená, že návrh je finální a může dojít ke konečnému schválení od výboru W3C Advisory Committee. Jakmile dojde ke schválení, z návrhu se stává doporučení a je ve finálním stavu Recommendation.

Aktuální stav modulů zahrnutých v CSS 3 je zobrazen na obrázku 1.




*Obrázek 1 Aktuální stav CSS 3 modulů. Krauss CSS3 taxonomy and status-v2.png [online] In:. [cit. 2018-03-20]. Dostupný pod licencí CC BY-SA 4.0 na www: https://commons.wikimedia.org/wiki/File:CSS3_taxonomy_and_status-v2.png*

Zelenou barvou jsou moduly ve stavu Reccomendation, žlutozelenou jsou ve stavu Candidate Recommendation a červeně ve stavu Working Draft.

Přínos CSS 3 je obrovský, protože to vývojářům přineslo mnoho nových možností, které doposud museli řešit jiným a neefektivním způsobem. Takovým příkladem můžou být oblé rohy, průhlednost, barevné přechody, stíny, animace atd.

## 1.2 Budoucnost CSS

Dalo by se předpokládat, že konscorcium W3C již pracuje na nové specifikaci CSS 4. Ovšem od zavedení modulů v CSS 3 již žádná nová verze kaskádových stylů nevznikne. Aktuálním cílem W3C je postupně vyvíjet aktuální specifikaci, případně ji rozšiřovat o nové funkce. Jednou se tedy má stát, že již nebudeme hovořit o kaskádových stylech verze 3, ale pouze o kaskádových stylech (Gasston, 2015, s. 30). V současné době již některé moduly, např. selektory, prošly čtvrtou iterací, což je v podstatě specifikace nové verze, ale stále to je a bude zařazeno do třetí verze kaskádových stylů.

Vývoj kaskádových stylů není tak rychlý, jak by si komunita přála, a proto dnes vznikají CSS preprocesory (více v samostatné kapitole), které se snaží kompenzovat jejich nedostatky. Vzhledem k jejich vzrůstající oblibě, si vývojáři začínají zvykat na prvky z programovacích jazyků, jako jsou proměnné a funkce. Dá se tedy očekávat, že v dlouhodobějším horizontu se kaskádové styly dočkají i této podpory nativně (Gasston, 2015, s. 215).

# 2 Aktuální trendy

Aktuální trendy ve stylování je možné pojmout ze dvou různých pohledů. První pohled je z pozice grafika a webdesignéra a druhý pohled je z pozice vývojáře a kodéra www stránek. Vývojář, potažmo kodér, není většinou příliš kreativní, a proto si musí udržovat přehled v aktuálně používaných technologiích a frameworcích. Zatímco grafik by měl být kreativní a držet se aktuálních grafických trendů, které především udávají velké společnosti jako je Google a Apple (Neumann, 2014). Tato bakalářská práce se spíše zaměřuje na pohled vývojáře, a proto si jen ve stručnosti popíšeme aktuální trendy posledních čtyř let z pohledu grafického návrhu.

## 2.1 Trendy z pohledu grafického návrhu

Vývoj webových stránek a obecně vývoj ve světě IT je velice rychlý a nemá smysl popisovat, jaké trendy byly v roce 2010 a dnes jsou již překonány. Avšak v roce 2015 byly nastoleny trendy, které více či méně přetrvávají dodnes.

**Trendy 2015**

Pro rok 2015 byl typickým znakem minimalismus, tzn. minimum textu, snaha o zdůraznění pouze podstatných informací, a to vše vyjádřeno velkým písmem (Václavek, 2015). Dalším znakem pro tento rok byl Flat design neboli plochý design. Tento design se zakládá na jednoduchosti a zabraňuje jakékoliv formě abstraktních ozdobných prvků, jako jsou přechody, reliéfy a stíny. Mezi další typické znaky je nutné zahrnout jednostránkové layouty, tzn. stránky, které nutí uživatele rolovat myší a veškerý obsah mají pod sebou.

Ovšem hlavním trendem pro tento rok byl responzivní webdesign (RWD). S tímto pojmem poprvé přišel v roce 2010 Ethan Marcotte pro magazín A List Apart (Sharkie, Fisher, 2015, s. 16). Hlavním cílem této metody je vytvořit pouze jednu variantu webové stránky, která se zobrazí na všech zařízeních stejně.

S responzivním webdesignem je úzce spjat termín mobile first. Mobile first je návrh responzivního webdesignu, který říká, že bychom měli nejprve navrhnout webové stránky pro mobilní zařízení a ty následně rozšířit na desktopovou verzi. Tato metoda je opakem běžného postupu, při němž se z desktopové verze následně pomocí media queries, nebo JavaScriptem odstraňují prvky, abychom získali vzhled pro mobilní zařízení.

**Trendy 2016**

V roce 2016 se spíše pokračuje v nastoleném trendu z roku 2015, který se doplňuje o nové prvky. Trendy www stránka v tomto roce obsahovala velký obrázek v hlavičce, byla široká přes celou obrazovku, obsahovala videa a v neposlední řadě se využívalo hodně CSS animací (WebDev, 2016). Webová stránka tedy měla obsahovat plynulé přechody mezi běžným stavem a hover stavem. Dále měly být všechny grafy interaktivní a moderním prvkem by bylo postupné zobrazování obsahu při rolování stránky dolů. Na oblibě také získal tzv. efekt Parallax. To je efekt, kde se zdá být poloha objektu odlišná při pohledu z různých pozic (Kozlerová, 2016). Posouvání stránky nám tedy dává iluzi, že dva objekty jsou v jedné přímce, ale pohybují se různou rychlostí.

**Trendy 2017**

Tento rok opět pokračoval v nastoleném trendu z předchozích let s důrazem na ještě větší minimalismus. Velkou oblibu získaly barevné gradienty (přechody), videa jako pozadí, výrazná typografie, experimentální kompozice prvků, mixování horizontálního a vertikálního umístění textu a vzájemné překrývání obrázků s textem. Velkým tématem bylo navigační menu a pokusy s jeho odstraněním, nebo netypickým umístěním. Příkladem může být užití pouze hamburger menu známého z mobilních verzí, nebo menu jako vyskakovací okno, popřípadě menu jako překrývající se okno (Brda, 2017).

**Trendy 2018**

Dá se očekávat, že se bude pokračovat v nastoleném trendu s důrazem na mobilní zařízení. Klíčový bude výběr použitých barev a písma. U barev můžeme očekávat pastelové odstíny s přechody a určitě nás mohou překvapit i layouty (rozložení webu), které budou nepravidelné, nesymetrické a doplněné ilustracemi (Kovařík, 2018).

## 2.2 Aktuální trendy z pohledu vývojáře a kodéra

Podpora hlavních modulů CSS 3 je v aktuálních verzích prohlížečů na dobré úrovni, a proto nic nebrání plně využívat všech možností, které nám umožňuje. Tato verze kaskádových stylů přináší velké množství nových a moderních vlastností a některé z nich budou detailněji ukázány v interaktivní prezentaci a popsány v kapitole věnované praktické části.

Ovšem dnešní vývojáři a především kodéři si jen s čistými kaskádovými styly příliš nevystačí. Zejména na větších projektech, a hlavně při práci v týmu, se využívá i dalších moderních metod. Těmi je použití frameworků, CSS preprocesorů a automatizačních nástrojů, které usnadňují a značně urychlují práci. V případě, že se webová stránka staví z jednotlivých komponent, je vhodné využít nějakého objektového přístupu a dodržovat konvence při pojmenovávání CSS tříd.

## 2.3 Objektově orientované a organizované psaní CSS

Konscoricum W3C neudává žádný standard, jakým by se mělo CSS zapisovat a používat. Proto se můžeme obzvláště u začínajících vývojářů setkat s poměrně chaotickým a nepřehledným zápisem. I každý pokročilý nebo profesionální vývojář používá jiný styl zápisu CSS pravidel. Systémů a stylů jakými psát pravidla je tedy nepřeberné množství, a proto téměř každá větší společnost provozující nějaký webový projekt vymyslela vlastní systém zápisu stylů, nebo rovnou celý CSS Framework, a ten následně poskytla široké veřejnosti. Při dnešním trendu psaní objektově orientovaných aplikací, vznikají i v CSS koncepty, které říkají, jak pracovat s objekty na webové stránce. Proto i starší návrhy jako je OOCSS, nebo SMACSS dnes získávají čím dál více na popularitě a každý profesionální kodér by o nich měl mít alespoň základní povědomí.

### 2.3.1 OOCSS

Object Oriented CSS je koncept psaní kódu, který v březnu 2009 prezentovala Nicole Sullivan (Michálek, 2017). Tento koncept se zaměřuje na psaní stylů pro jednotlivé komponenty neboli CSS objekty. Pod komponentou si můžeme představit část HTML kódu s kaskádovým stylem, případně i JavaScriptem. Principem této metody je přemýšlet o webové stránce jako o samostatných částech (komponentech), které se mohou na dané stránce opakovat, nebo ještě lépe i v úplně jiné stránce. Použitím této metody si kromě možnosti znovupoužití kódu můžeme zjednodušit i jeho spravovatelnost. Nejjednodušším příkladem komponenty na www stránce může být tlačítko.

```
<button></button>
```

Standardně bychom tento prvek stylovali např. takto:

```
button { color: white; background: black; padding: 1em; }
```

Ovšem v případě použití OOCSS přístupu je tento postup chybný. Správně je, když se danému elementu přidá CSS třída např. .button, takže výsledný HTML kód bude vypadat následujícím způsobem:

```
<button class="button"></button>
```

Teď by bylo možné nastylovat třídu button a dát jí vlastnosti z příkladu výše, tzn. color, background a padding. Tento postup je opět nevhodný. Pravděpodobně se na stránce bude vyskytovat tlačítek více. Je tedy vhodné nastavit pouze výchozí styl, jako je např. padding, a následně přidat a nastylovat další třídu. Např. jako tlačítko primary a secondary.

```
<button class="button button-primary"></button>
<button class="button button-secondary"></button>

.button { padding: 1em; }
.button-primary { color: white; background: black; }
.button-secondary { color: black; background: green; }
```

Pokud budeme chtít přidat nějakou další vlastnost, přiřadíme elementu další třídu a tu nastylujeme.

Tento koncept ovšem nemá pevně stanové pravidla, ale je to spíše sada doporučení, jak pracovat v CSS s objekty. Proto můžeme najít různé principy a doporučení, které se mohou lišit vývojář od vývojáře.

Např. pojetí OOCSS v podání Martina Michálka má pět principů (Michálek, 2017).

1. Nezávislost vzhledu na struktuře.
    - Tento princip je možné jednoduše vysvětlit tímto způsobem: do CSS selektorů se nikdy nedávají HTML tagy. Tzn. nebudeme stylovat element button, ale vytvoříme zvlášť třídu .button.
2. Nezávislost obsahu na kontejneru.

    - Nezávislost obsahu na kontejneru říká, že CSS selektory nebudou obsahovat strukturu HTML. Pokud máme element button umístěn např. ve formuláři, nebudeme tlačítko stylovat jako .form .button, ale pouze samostatně jako .button.
3. Co nejnižší specifičnost.
    - Tento bod se váže na předchozí, kdy se snažíme mít element specifikován co nejjednodušeji. To se ovšem vylučuje s používáním ID selektorů, nebo použitím klauzule !important. Z tohoto důvodu je použití obojího v OOCSS vyloučeno.
4. Vývoj zaměřený na komponenty a znovupoužitelnost.
5. Princip prefixování.
    - Michalík uvádí tři typy prvků a dle toho styl zápisu pravidel:
        - objekt – komponenta (.button),
        - element – prvek uvnitř elementu (.button-icon),
        - modifikátor – vlastnost objektu (.button-primary)

K podobnému závěru došel i Patrik Illy, který měl v roce 2014 prezentaci na toto téma na ostravském srazu Frontendistů (Illy, 2014). Ten uvádí principů sedm, ale jsou téměř totožné s principy, které uvádí Michalík.

1. Přemýšlet v objektech, ne ve stránkách.
2. Rozšiřovat objekt použitím více tříd.
3. Oddělovat strukturu od vzhledu.
4. Vyhnout se závislosti na kontextu.
5. Používat mřížku.
6. Nepoužívat ID selektory.
7. Nesvazovat selektor s konkrétním elementem.

V podstatě jediný větší rozdíl je v pátém principu používání mřížky. Zde je ovšem dosti zavádějící, zda tento princip řadit mezi principy OMCSS. Pod tímto principem si Illy představuje umístění komponenty do nějaké mřížky a tím pádem se zaručí její optimální zobrazení.

Mezi hlavní výhodu tohoto konceptu patří možnost znovupoužití komponent a naopak mezi nevýhody je možné zařadit, že pro změnu vzhledu komponenty je nutné zasahovat do HTML a přidávat další třídy. Další nevýhodou může být, že po nahlédnutí do CSS souboru není hned jasné, jak daná komponenta vypadá, protože bez pohledu na HTML není možné zjistit, jaké má komponenta třídy.

### 2.3.2 SMACSS

Scalable and Modular Architecture for CSS je kniha od Jonathana Snooka, která je metodikou psaní kaskádových stylů. Hlavním principem této metody je uspořádat a organizovat psaní CSS za pomoci kategorizace a logického pojmenování tříd dle kategorie (Snook, 2012, s. 4). Jonathan Snook proto přichází s rozdělením kaskádových stylů do pěti kategorií a každou z nich detailně popisuje, co by měla obsahovat.

Tyto kategorie se nazývají:

1. Base
2. Layout
3. Module
4. State
5. Theme

**Base**

Základní kategorie slouží k výchozímu nastavení HTML elementů i jejich pseudotřídám. Jako příklad těchto elementů je možné uvést např. a, a:hover, form a body (Snook, 2012, s. 8). Dále do této kategorie patří importování písma, importy externích stylů, nastavení velikosti záhlaví atd. Měl by se tu vyskytovat i tzv. Reset CSS, to je sada základních stylů, které slouží pro odstranění, nebo obnovení výchozích CSS nastavení za účelem dosažení konzistentního zobrazení napříč všemi webovými prohlížeči.

Příklad, jak by mohl soubor base.css vypadat:

```
* { box-sizing: border-box; }
body, form { margin: 0; padding: 0; }
a { color: black; }
a:hover { color: red; }
```

**Layout**

Kategorie rozložení slouží k umístění prvků na stránce. To si je možné představit tak, že tento styl obsahuje třídy pro manipulaci s obsahem, případně přímo návrh grid tabulky (zobrazení v mřížce). Všechny třídy definované v této kategorii by měly být jasně rozpoznatelné, a proto je vhodné jim přidat nějaký prefix. Přímo Snook pracuje s prefixem l- (Snook, 2012, s. 12).

Příklad souboru layout.css:

```
.l-grid {…}
.l-grid > li {…}
.l-left { float: left; }
.l-right { float: right; }
.l-align-center { text-align: center; }
```

**Module**

Moduly jsou opakovaně použitelné modulární části designu (Snook, 2012, s. 5). Příkladem takového modulu může být navigační lišta, dialogy, nebo formuláře. Pokud jsou moduly napsány správně, nic nebrání jejich použití i v kategorii rozvržení, nebo dokonce v jiném projektu. Na rozdíl od předchozí kategorie, moduly neobsahují žádný prefix.

Do souboru module.css se tedy zapisují všechny moduly a optimálně jako import souborů s daným modulem ze složky module.

Soubor nav.css ve složce module:

```
.nav { width: 100%; padding: 0; }
.nav > h2 { padding: 5px; }
```

Module.css:

```
@import "module/nav.css";
@import "module/loginForm.css";
```

```
@import …
```

**State**

Tato kategorie se věnuje rozšíření ostatních stylů a především modulů o stavy. To znamená, že slouží k nějaké změně. Nejčastěji se tak stane po interakci uživatelem pomocí JavaScriptu, nebo po aplikovaní media queries pravidla. Zde Snook opět zavádí prefix a je jím is- (Snook, 2012, s. 6).

Použití je možné ukázat např. na předchozím modulu nav.css, který je rozšířen o dva stavy:

```
.nav { width: 100%; padding: 0; }
.nav > h2 { padding: 5px; }

.nav.is-collapsed { background-color: black; }
.nav.is-opened { background-color: white; }
```

**Theme**

Téma je samostatná kategorie, potažmo CSS soubor, který slouží ke změně vzhledu. Použitím tohoto předpisu je umožněno vytvářet alternativní témata vzhledu a je možné jím měnit i stavy.

Příklad si je možné uvést např. na modulu menu.css:

```
.menu { border: 1px solid red; }
.menu > h3 { color: red; }
```

Soubor theme.css tedy bude vypadat třeba takto:

```
.menu { border: 2px dashed green; }
```

Hlavní výhodou této metodiky je logické členění souborů, zavedení prefixů a práce s moduly, které umožňují jednou napsané styly sdílet mezi ostatními stránkami nebo projekty.

### 2.3.3 BEM

Je plnohodnotná metodika pro organizaci a psaní CSS navržená ruským vyhledávačem Yandex. Název BEM je zkratkou slov Block, Element, Modifier a je to přístup obdobně zaměřený jako OOCSS na vývoj komponent webové stránky. Vznikl v březnu 2009 jako součást Lego 2.0 frameworku, který používali v Yandexu. V roce 2010 byla založena organizace BEM a z frameworku Lego byla vyjmuta část zabývající se bloky do samostatné BEM knihovny. Ovšem BEM dnes není jen metodika od Yandexu, ale je to především konvence pojmenovávání tříd, kterou z původní metodiky převzali další autoři. Tato konvence se může velice pěkně doplňovat se systémy organizace jako je SMACSS (Michalík, 2017).

Princip BEMu spočívá v pojmenovávání komponent, kde se zcela vylučuje použití ID selektorů a všechny částí komponenty obsahují svoje třídy. Z tohoto důvodu je tato metodika vhodnější na větší projekty a pro práci v týmu. Celá metodika se opírá o tři hlavní pojmy, které je nutné pochopit.

**Blok**

Základní prvek BEMu je blok, tím se rozumí nějaká komponenta na webové stránce. Tento blok je nezávislý prvek, který je znovupoužitelný a umožňuje do sebe libovolně vnořovat bloky jiné. Příkladem takového bloku může být vytvoření komponenty „moje-komponenta“ s bílým pozadím.

```
<div class="moje-komponenta">
    ...
</div>
```

```
.moje-komponenta { background-color: #fff; }
```

Jak již bylo uvedeno výše, každá komponenta má vlastní třídu a pokud je nutné, je oddělena spojovníkem: .nazev-bloku.

**Element**

Je prvek uvnitř bloku, který chceme stylovat a jeho existence má smysl jen v rámci tohoto bloku. Je tedy samostatně nepoužitelný a zapisuje se stylem .nazev-bloku__nazev-elementu. Opět si ukažme příklad:

```
<div class="moje-komponenta">
  <a href="#" class="moje-komponenta__odkaz">Odkaz</a>
  <a href="#" class="moje-komponenta__odkaz">Odkaz</a>
</div>
```

```
.moje-komponenta {…}
.moje-komponenta__odkaz { color: #fff; }
```

Tento příklad dobře demonstruje pojmenování elementu, který je přímý potomek bloku. Ovšem poměrně často se setkáme se situací, kde element je potomkem jiného elementu. Opět si uvedeme příklad:

```
<div class="moje-komponenta">
  <div class="moje-komponenta__header">
    <h2 class="moje-komponenta__title">Nadpis</h2>
  </div>
</div>
```

Ačkoliv by se tu nabízelo použít pro nadpis třídu .moje-komponenta__header__title, je tento přístup nevhodný. Už proto, že by se selektory staly nesnesitelně dlouhé a taky proto, že při změně HTML by se musel měnit i CSS selektor (Berner, 2016). Proto je v tomto případě vhodnější nerespektovat označení dle DOM struktury a použít označení elementu jako .moje-komponenta__nazev-elementu.

**Modifikátor**

Umožňuje vytvořit variantu komponenty i elementu. Jinak řečeno slouží ke změně stavu, chování, nebo popisuje nějakou vlastnost. Zapisuje se jako třída prefixovaná názvem bloku, nebo elementu se dvěma pomlčkami, takže: nazev-bloku—modifikator.

```
<div class="moje-komponenta moje-komponenta--secondary">
  <a href="#" class="moje-komponenta__odkaz moje-
komponenta__odkaz--active">Odkaz</a>
  <a href="#" class="moje-komponenta__odkaz moje-
komponenta__odkaz--hide">Odkaz</a>
</div>
```

```
.moje-komponenta { background-color: red; }
.moje-komponenta—secondary { background-color: blue; }
.moje-komponenta__odkaz { color: white; }
.moje-komponenta__odkaz--active { border: 1px solid green;}
.moje-komponenta__odkaz--hide { display: none; }
```

BEM metodologie je dobrý způsob při zapisování stylů a obzvláště silnou zbraní je při použití s CSS preprocesory. Podobně jako v SMACSS je vhodné pro každou komponentu vytvořit extra CSS soubor a tím dosáhnout modulárnosti a znovupožitelnosti. Nevýhodou jsou dlouhé názvy tříd a nutnost každému elementu přiřadit nějakou třídu.

### 2.3.4 Atomic CSS

Dnes kromě metodik a frameworků je možné využít i dalších způsobů práce se styly. Příkladem je Atomic CSS, který rozhodně není možné zařadit mezi výše uvedené metodiky. Atomic se nezabývá modulárním přístupem, ani nemá standardní syntaxi jazyka CSS, ale je to nástroj, který generuje CSS na základě definic obsažených v HTML. Takže místo toho, aby se daným HTML elementům přiřadila CSS třída, v Atomicu se zapíší přímo styly do atributu class. Následně prekompilátor s názvem Atomizer projde HTML soubor a vytvoří třídy s jejich styly.

Příklad takového HTML elementu s použitím Atomic CSS může vypadat třeba následujícím způsobem:

```
<div class="Bgc(#000) C(#fff.5) P(10px) Bd">
  Obsah
</div>
```

Výsledný CSS soubor pro výše uvedený kód vypadá takto:

```
.Bgc\(\#000\) { background-color: #000; }
.C\(\#fff\.5\) { color: rgba(255,255,255,.5); }
.P\(\20px\) { padding: 20px; }
.Bd { border-style: solid; }
```

Jako další příklad si můžeme uvést tento kód, který nám po najetí myši na element div obarví odstavec na bílou barvu z původní šedé.

```
<div class="foo Bgc(#000)">
  <p class="Bgc(#e2e2e2) foo:h>Bgc(#fff)">Obsah</p>
</div>
```

Konstrukce jazyka je značně odlišná od jazyka CSS, což může pro někoho být výhoda i nevýhoda. Výhodou Atomicu je možnost definování si proměnných, které je možné použít v dokumentu. Ovšem největší nevýhodou Atomicu je nutnost měnit HTML soubor při každé změně stylu.

## 2.4 CSS frameworky

Definic, které obecně vysvětlují pojem framework, je nepřeberné množství. Jednou z takových definic může být: *„Framework je softwarová struktura, která slouží jako podpora při programování a vývoji a organizaci jiných softwarových projektů. Může obsahovat podpůrné programy, knihovny API, podporu pro návrhové vzory nebo doporučené postupy při vývoji.“* (Wikipedia). Tato definice spíše formuluje pojem framework v programovacích jazycích jako je Java, C++ nebo C#. Proto si můžeme uvést ještě další obecnou definici, která bude více odpovídat webovým technologiím *„Framework je standardizovaný soubor konceptů, postupů a kritérií pro řešení běžných typů problémů, které mohou být použity jako doporučení pro přistupování a řešení nových problémů podobného charakteru.“* (Awwwards). Ve světě webových technologií ve smyslu kaskádových stylů je chápání frameworku ještě v daleko užším smyslu a lze jej definovat jako *„balíček předpřipravených souborů a složek standardizovaného kódu, které jsou*

*využity při vývoji webových stránek, nebo jako základ při tvorbě nového webu*“ (Awwwards).

Z výše uvedené definice tedy vyplývá, že CSS framework je ucelená knihovna, nebo sada knihoven, která dává dohromady hotové řešení a značným způsobem může urychlit a zjednodušit stylování webové stránky. CSS frameworky je možné rozdělit na jednoduché a pokročilé (Awwwards). Jednoduché jsou mnohdy jen grid systémy. To znamená, že umožňují umístění všech elementů webové stránky do pravidelné mřížky. Nespornou výhodou tohoto systému je, že je připraven na responzivní variantu zobrazení, což velice usnadňuje práci. Mezi takové jednoduché frameworky je možné zařadit 960 Grid System[1] nebo Base[2]. Naopak pokročilé frameworky kromě grid systému obsahují i předstylované komponenty např. tlačítka a další funkce pomocí JavaScriptu. Proto se spíše obecněji řadí mezi front-end frameworky. Mezi nejznámější představitele této kategorie je možné zařadit Bootstrap[3], Foundation[4] a W3.CSS[5].

Výhoda frameworků tedy spočívá v tom, že je to hotový funkční celek, který je možné jednoduše používat napříč projekty a není nutné se zabývat vývojem vždy od nuly. Díky předpřipraveným komponentám může i odpadnout práce na grafickém návrhu. To ocení obzvláště programátoři bez grafického cítění, kteří mohou jednoduše vytvořit pěkný a moderní vzhled. Framework není nic složitého a každý vývojář si může napsat svůj vlastní. Avšak předností již hotového frameworku, který v lepším případě vyvinula nějaká větší společnost, má nespornou výhodu v tom, že je to odladěný a funkční celek. Samozřejmě je tu riziko, že vývoj bude ukončen, nebo že novější verze nebude zpětně kompatibilní. Obecně největším úskalím všech těchto systémů je nutnost naučení a pochopení jeho principů. Dnes prakticky každá větší webová stránka je postavená na nějakém frameworku a je povinností vývojářů a kodérů, aby byli seznámeni s alespoň těmi nejpoužívanějšími.

[1] Dostupné z: http://getbase.org/
[2] Dostupné z: https://960.gs/
[3] Dostupné z: https://getbootstrap.com/
[4] Dostupné z: https://foundation.zurb.com/
[5] Dostupné z: https://www.w3schools.com/w3css/

### 2.4.1 Bootstrap

Bootstrap patří mezi nejznámější a nejoblíbenější frontend frameworky současnosti. Jeho oblíbenost potvrzuje obdržený počet více jak 123 tisíc hvězdiček na GitHubu. Což je 3x tolik než druhý nejoblíbenější framework Semantic-UI. Bootstrap vznikl ve společnosti Twitter v polovině roku 2010 pod názvem Twitter Blueprint (Bootstrap). Důvod, proč došlo k jeho vzniku, byl ten, že zaměstnance společnosti trápila nekonzistence interních aplikací ve firmě. Jejich vzhled byl značně odlišný a využívalo se různých externích knihoven, což vedlo k velké náročnosti na údržbu kódu. Proto se vývojáři Twitteru Mark Otto a Jacob Thornton rozhodli v rámci firemních hackathonů vymyslet knihovnu, která sjednotí kód všech jejich aplikací. V srpnu 2011 byl projekt přejmenován na Bootstrap a uveřejněn jako open source projekt na GitHubu.

Dne 31. ledna 2012 byla vydána verze Bootstrap 2, která přinesla podporu responzivního designu. A nejen kvůli tomu se v únoru stal Bootstrap nejoblíbenějším projektem na GitHubu (Otto, 2012). Kromě podpory responzivního designu přinesl nový 12 sloupcový layout podporu ikonek od Glyphicons a celkově vylepšené komponenty.

Další verze Bootstrap 3, která vyšla v srpnu 2013 a přinesla nový flat design, se zaměřuje na mobile first návrh, přidává nové komponenty a přestává podporovat Internet Explorer 7 a Firefox 3.6 (Sitepoint, 2013). Došlo i ke změně licence, kterou je nyní MIT. Vývoj aktuálně poslední verze 4 byl oznámen v říjnu 2014 (Otto, 2014) a finálně byla vydána v lednu 2018 (Otto, 2018).

Poslední verze přinesla opět spoustu novinek. Zásadní změnou je, že grid systém již není postaven pomocí float, ale pomocí flexbox. Ten není podporován v Internet Exploreru 8 ani 9 a proto webové stránky postavené na Bootstrap 4 budou fungovat pouze v novějších prohlížečích. Další podstatnou změnou je, že Bootstrap, který byl do té doby postavený na CSS preprocesoru LESS, nyní přechází na SASS.

Bootstrap je založený na principu OOCSS. To znamená, že je modulární a tedy složený z jednotlivých částí a není nutné všechny používat. Na webové stránce, na které se využívá pouze grid systém, je zbytečné, aby byly přidávány i moduly s předstylovanými komponentami, které se nevyužívají. V tomto ohledu je vhodné Bootstrap používat s preprocesory a importovat jednotlivé moduly. Samozřejmě je možné jej používat i

bez preprocesorů, jelikož na oficiálních webových stránkách je možné stáhnout již vygenerovaný celý Bootstrap jako jeden CSS soubor. Autoři tohoto frameworku počítali s tím, že ne každý využije všechny jeho možnosti, a proto je možné stáhnout kromě kompletního Bootstrapu pouze moduly bootstrap-grid a bootstrap-reboot (platí pro Bootstrap ve verzi 4).

Aby mohly být využity všechny možnosti, které Bootstrap přináší, je nutné kromě CSS souborů do webové stránky přidat i knihovnu napsanou v JavaScriptu. Tato knihovna není napsána v čistém JavaScriptu, ale je závislá na knihovně jQuery a Popper.

Ve čtvrté verzi přišel Bootstrap s vlastním resetovacím stylem, který pojmenoval Reboot. Ten je postavený na projektu Normalize[6], který si klade za cíl nastavit optimální výchozí styly napříč všemi webovými prohlížeči za účelem snazšího stylování. Reboot upravuje výchozí styl elementů bez nutnosti přidat k nim jakékoli CSS třídy. Kupříkladu všem elementům přidává CSS vlastnost box-sizing: border-box, čímž se velikost elementů vždy počítá včetně rámečku, paddingu a marginu. Dalším nastavením je např. změna výchozího písma, jeho barva, velikost, řádkování, odebrání okrajů u tabulky, nebo i barva pozadí dokumentu.

Obecně princip fungování těchto frontend frameworků spočívá v přidávání určitých CSS tříd HTML elementům. Každý větší a známější framework má svoji oficiální dokumentaci, kde je uvedeno, co vše umí a jaké třídy jsou připravené. Příkladem mohou být tlačítka, která mají obecnou třídu .btn a několik dalších variant, které se liší účelem a jsou barevně odlišené.

```
<button class="btn btn-primary">Primary</button>
<button class="btn btn-secondary">Secondary</button>
<button class="btn btn-success">Success</button>
<button class="btn btn-danger">Danger</button>
<button class="btn btn-warning">Warning</button>
<button class="btn btn-info">Info</button>
<button class="btn btn-light">Light</button>
<button class="btn btn-dark">Dark</button>
```

[6] Dostupné z: https://necolas.github.io/normalize.css/

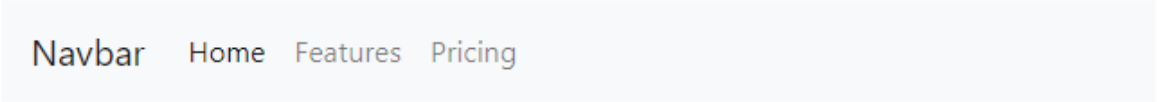

*Obrázek 2 Bootstrap 4 varianty tlačítek*

Třídy nejsou pevně vázány na konkrétní HTML element, a je tedy možné třídu .btn přidat i k jinému elementu, než je button.

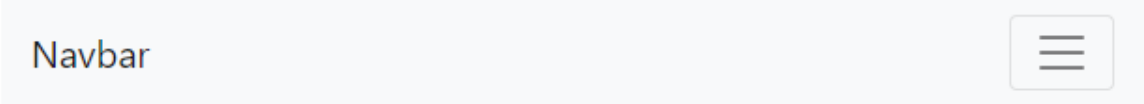
Protože je Bootstrap responzivní framework, má i předpřipravené navigační menu, které se přizpůsobuje zařízení. To znamená, že se automaticky přepíná do mobilního zobrazení, tzv. hamburger varianty. Příklad HTML kódu s CSS třídami je možné nalézt v oficiální dokumentaci Bootstrapu a může vypadat následujícím způsobem:

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">

  <a class="navbar-brand" href="#">Navbar</a>

  <button class="navbar-toggler" type="button" data-
toggle="collapse" data-target="#navbarNav" aria-
controls="navbarNav" aria-expanded="false" aria-
label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>

  <div class="collapse navbar-collapse" id="navbarNav">
    <ul class="navbar-nav">
      <li class="nav-item active">
        <a class="nav-link" href="#">Home</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Features</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Pricing</a>
      </li>
    </ul>
  </div>

</nav>
```

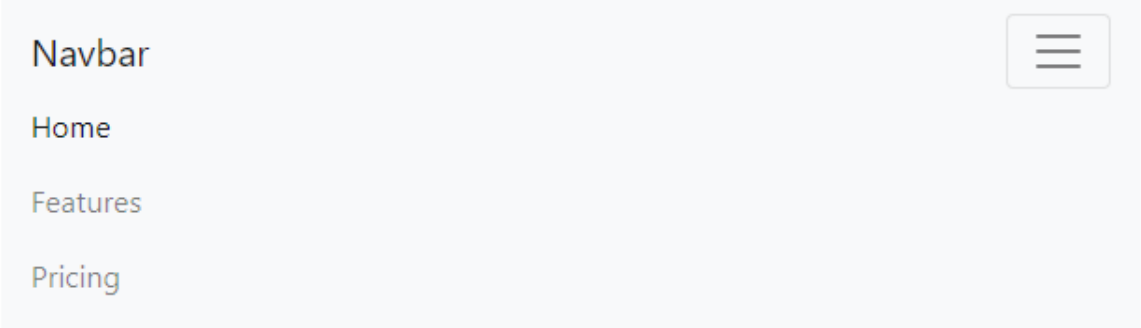
Výsledné menu se poté může vyskytovat ve třech stavech. První stav na obrázku 3 je klasické menu na běžném monitoru. Druhý stav je na obrázku 4, kde je menu zobrazené na mobilním zařízení. Poslední stav je zobrazen na obrázku 5, kde se menu rozbalí kliknutím na hamburger tlačítko. Celá funkčnost je založená na pouhém doplnění tříd a atributů několika HTML elementům, což by bez použití frameworku znamenalo mnohem více práce.




*Obrázek 3 Bootstrap 4 navbar*




*Obrázek 4 Bootstrap 4 navbar hamburger menu*

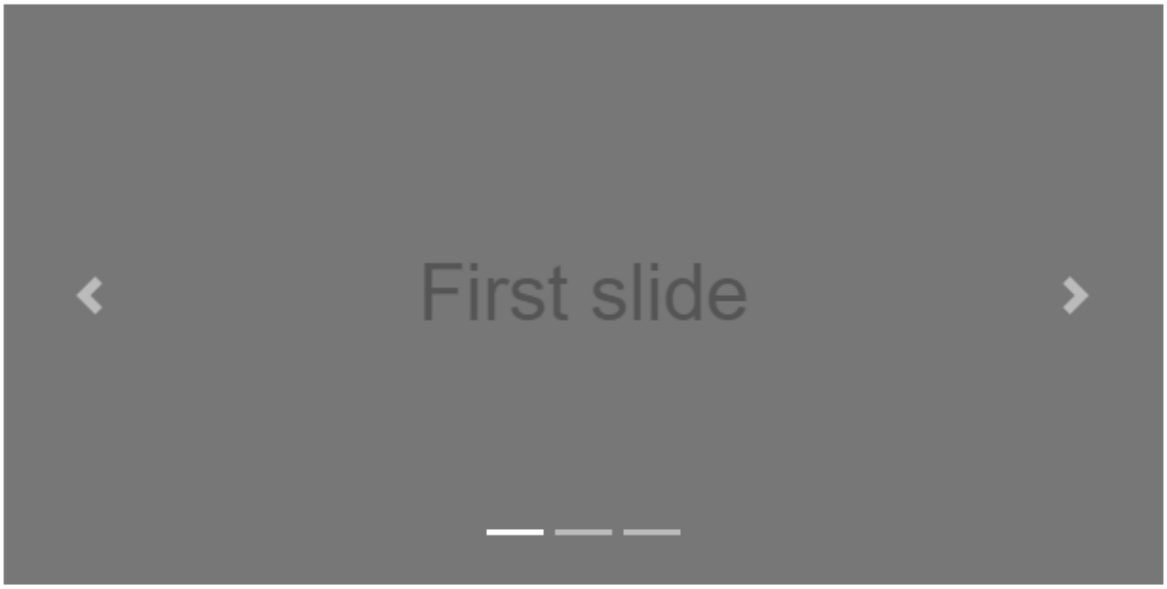



*Obrázek 5 Bootsrap 4 navbar rozbalené hamburger menu*

Dalším příkladem již pokročilejších funkcí, které jsou zcela závislé na JavaScriptové knihovně, může být komponenta Carousel. To je dnes poměrně hodně častý prvek na webových stránkách, který vytváří slideshow obrázků nebo textů. Opět poměrně složitá funkce, která znamená psaní velkého množství kódu, je pomocí Bootstrapu otázkou pár HTML atributů. Kód takového slideshow opět nalezneme v dokumentaci a vypadá třeba takto:

```
<div id="carousel" class="carousel slide" data-
ride="carousel">

  <ol class="carousel-indicators">
```

```
    <li data-target="#carousel" data-slide-to="0"
class="active"></li>
    <li data-target="#carousel" data-slide-to="1"></li>
    <li data-target="#carousel" data-slide-to="2"></li>
  </ol>

  <div class="carousel-inner">
    <div class="carousel-item active">
      <img src="obrazek.jpg">
    </div>
    <div class="carousel-item">
      <img src="obrazek2.jpg">
    </div>
    <div class="carousel-item">
      <img src="obrazek3.jpg">
    </div>
  </div>

  <a class="carousel-control-prev" href="#carousel"
role="button" data-slide="prev">
    <span class="carousel-control-prev-icon" aria-
hidden="true"></span>
    <span class="sr-only">Previous</span>
  </a>

  <a class="carousel-control-next" href="#carousel"
role="button" data-slide="next">
    <span class="carousel-control-next-icon" aria-
hidden="true"></span>
    <span class="sr-only">Next</span>
  </a>

</div>
```




*Obrázek 6 Bootstrap 4 ukázka Carousel komponenty*

Další JavasSriptové komponenty jsou např. tooltipy (bublinkové nápovědy), pop-over (otevírací okna), Dropdawn (rozbalovací políčka s nabídkou) a další.

Základem všech těchto frameworků je tzv. grid systém. Předchůdce tohoto systému byl tabulkový layout (rozložení), což je v dnešní době už velký přežitek. Ať už proto, že tabulky nejsou sémantické, tak hlavně i proto, že nejsou responzivní. Z tohoto důvodu musel vzniknout jiný způsob, který umožní vytvářet pravidelnou a responzivní mřížku. Využívají se na to dva způsoby. Prvním je zarovnání pomocí CSS vlastnosti float. Tento způsob má výhodu, že je podporován i ve starších prohlížečích, ale má i své nevýhody a určitá omezení, například nemožnost mít stejně vysoké elementy s float vlastností, nebo provádět vertikální zarovnání. Proto v aktuální verzi Bootstrapu se již využívá flexbox vlastností, které uvedenými nevýhodami netrpí. Grid systémy obvykle fungují tak, že se elementům přidají předpřipravené třídy, které tak získají určitou šířku. Bootstrap standardně pracuje s 12 sloupci, protože číslo 12 má mnoho dělitelů (Čápka). Je tedy možné vytvořit layout přes celou šířku (12 sloupců), přes polovinu (6 sloupců), přes třetinu šířky (4 sloupce), přes čtvrtinu (3 sloupce), přes 1/6 (2 sloupce) a přes 1/12 šířky (1 sloupec). Responzivita gridu spočívá v zalomení sloupečku v případě, že se na stránku nevejde, nebo v předem stanovené

šířce. Tyto frameworky pracují s přednastavenými CSS třídami, které mají připravené breakpointy při určité šířce. Bootsrap 4 má celkem pět breakpointů.

*Tabulka 1 Bootstrap breakpointy*

| Název | Extra small | Small | Medium | Large | Extra large |
|---|---|---|---|---|---|
| Šířka | < 576px | ≥ 576px | ≥ 768px | ≥ 992px | ≥ 1200px |
| CSS třída | .col- | .col-sm- | .col-md- | .col-lg- | .col-xl- |

Příklad použití:

```
<div class="row">
  <div class="col-12 col-sm-8">.col-12 .col-md-8</div>
  <div class="col-6 col-sm-4">.col-6 .col-md-4</div>
</div>
```

První div element má povinně třídu .row, která vytváří kontejner pro sloupečky a je na rozdíl od předchozích verzí v Bootstrapu 4 nutná pro funkčnost. Vnořené elementy s třídou .col-* v kontejneru .row nám vytváří sloupečky. V ukázce výše jsou vytvořené dva div elementy, takže se vytváří dva sloupce. První element má třídu .col-12, která udává, že jakmile šířka bude nižší jak 576px nastaví sloupeček na maximální šířku tzn. 12 sloupečků. Zatímco druhý element při této šířce skočí na druhý řádek a bude zabírat poloviční šířku. Třída .col-sm-8 udává, že při šířce větší jak 576px bude sloupeček zabírat 8/12 místa a druhý bude zabírat 4/12 místa. Takže oba sloupečky budou vedle sebe.


*Obrázek 7 Bootstrap 4 grid při šířce menší než 576px*

*Obrázek 8 Bootstrap 4 grid při šířce větší než 576px*

Používání Bootstrap gridu je jednoduché a hlavně velice užitečné. Při dnešním vývoji webové stránky je to nesporný pomocník, a pokud z jakéhokoli důvodu nemají být na webové stránce využity všechny komponenty Bootstrapu, grid a reset modul je vhodné použít jako základ snad každé www stránky. Výhoda Bootstrapu je ve velké komunitě, rozsáhlé dokumentaci a v aktivním vývoji. Mezi nevýhody je možné zařadit velikost souborů a nadměrný počet CSS tříd.

### 2.4.2 Foundation

Dalším oblíbeným frontend frameworkem je Foundation. Ten má kořeny již v roce 2008, kdy společnost ZURB používala pro klientské weby předpřipravené styly. O tři roky později byl tento framework vydán jako open source projekt pod licencí MIT a byl to první framework, který podporoval responzivní design a mobile first přístup (Foundation). Stejně jako Bootstrap je modulární a je postavený na CSS preprocesoru SASS. Na rozdíl od něj má pouze základní design komponent, které se poté snadněji stylují. Také grid systém je lehce odlišný, Foundation v poslední verzi používá pouze tři základní breakpointy (Foundation for Sites 6).

*Tabulka 2 Foundation breakpointy*

| Název | Small | Medium | Large |
|---|---|---|---|
| Šířka | jakákoliv | ≥ 640px | ≥ 1024px |
| CSS třída | .small- | .medium- | .large- |

Foundation pracuje na stejném principu jako Bootstrap, tzn. pomocí přidávání CSS tříd HTML elementům. I některé komponenty taktéž vyžadují JavaScriptovou jQuery knihovnu. Oproti Bootstrapu má ještě o pár kilobajtů více a je náročnější k naučení. Je tedy vhodný spíše pro zkušené vývojáře, kteří se chtějí zabývat rychlým vývojem webových stránek (KeyCDN, 2018).

### 2.4.3 W3.CSS

CSS potažmo frondend frameworků je nepřeberné množství a jedním z nich je i W3.CSS. Z názvu by se mohlo zdát, že se jedná o framework přímo od konscorcia W3C, ale není tomu

tak. Za vývojem tohoto frameworku stojí známý vývojářský web W3Schools. Tento framework se neřadí mezi nejoblíbenější ani nejpoužívanější, ale obsahuje v sobě všechny podstatné funkce a moduly. Stejně jako výše zmíněné frameworky je bezplatný, responzivní a preferuje mobile first přístup. Na rozdíl od těchto frameworků využívá pouze CSS vlastností a neobsahuje žádnou JavaScriptovou knihovnu. Díky tomu je rychlejší, menší a jednodušší k naučení (W3Schools). Obsahuje několik připravených barevných témat, jako jsou barvy pro flat design, nebo material design. Prefix CSS tříd tohoto frameworku vždy začíná w3-, což na někoho působí ve zdrojovém kódu značně nepřehledně.

## 2.5 Preprocesory a postprocesory

Pojem CSS preprocesor již padl v souvislosti s frameworky. Je to v podstatě jazyk, který je postavený nad CSS (Michálek, 2017). Preprocesory přidávají nové vlastnosti, nebo řeší technické nedostatky kaskádových stylů. Vznikly jako reakce na zanedbaný a pomalý vývoj CSS.

Pokud se píše kód pro preprocesor, píše se jakýsi mezikód, který je nutné přeložit do cílového jazyka (Sharkie, Fisher, 2015, s. 135). To znamená, že se využívá mezilehlý formát souborů a syntaxe dle zvoleného preprocesoru. Napsaný kód se posléze převede na standardní kód jazyka CSS. Tímto způsobem z čistě deklarativního jazyka, děláme jazyk kompilovaný, což přináší spoustu výhod (Javorek, 2011).

Výhod používání preprocesorů oproti čistému CSS kódu je hned několik a čím větší projekt se vytváří, tím výhody této technologie rostou. Martin Michalík uvádí pět zásadních důvodu proč preprocesory používat (Michalík, 2014).

- Spravovatelnost
    - Preprocesory v kombinaci s nějakou metodou organizace kódu jsou velice užitečné. Např. v kombinaci s BEM metodikou je možné využít zanořování.
- Organizace
    - Výhodou je možnost organizovat kód přes moduly, malé kousky kódu vztažené ke konkrétní komponentě.
- Rychlejší psaní kódu a změny v něm
    - Rychlejší vývoj díky využití mixinů a proměnných v preprocesorech.

- Využití hotových knihoven
    - Díky preprocesorům vznikají hotové knihovny nebo frameworky.
- Vlastní knihovny
    - Jednoduchá možnost, jak si napsat vlastní framework.

Nevýhodou preprocesorů je nutnost vykonat krok navíc, a tím je samotná kompilace do CSS. I na tento krok se v dnešní době myslí, a proto existuje spousta automatizovaných nástrojů, které provádí kompilaci automaticky, nebo je možno využít kompilace pomocí JavaScriptu přímo ve webovém prohlížeči při načítání stránky. Tato možnost je vhodná pouze během testování a vývoje. Pro reálné použití na www stránce je lepší mít již zkompilovaný CSS soubor. Mezi další nevýhody patří i učení se novému jazyku.

Nejznámější a nejpoužívanější preprocesory jsou Sass, Less a Stylus. Mezi výše zmíněnými preprocesory je malý rozdíl a přejít z jednoho preprocesoru na druhý je poměrně bezproblémová záležitost.

Oproti preprocesorům existují ještě tzv. postprocesory. Nejznámějším zástupcem této kategorie je PostCSS. Rozdíl oproti preprocesorům spočívá v momentě zpracování. Zatímco preprocesory předzpracovávají, což znamená, že se v podstatě vytváří nový jazyk, který se následně kompiluje, tak postprocesory počítají se současnou, nebo budoucí verzí CSS a provádějí následnou transformaci pomocí JavaScriptu. Samotný PostCSS je spíše takové API pro jeho moduly. Příkladem takového modulu je třeba autoprefixer. Tento modul analyzuje náš CSS kód a přidává prefix pravidlům.

```
.example {
  color: red;
  transition: all .3s;
}
```

Výše uvedený kousek CSS pravidel, se automaticky upraví na výstup s prefixy.

```
.example {
  color: red;
  -webkit-transition: all .3s;
  transition: all .3s;
```

```
}
```

V současné době jsou preprocesory hojně využívané a je nutné mít alespoň základní přehled o jejich fungování.

### 2.5.1 SASS

Syntactically Awesome Style Sheets je CSS preprocesor, který vznikl již v roce 2006 (Spławski, 2015). Jeho cílem je rozšířit CSS o proměnné, cykly, podmínky, mixiny a další funkce. Sass má dva druhy syntaxe, první a starší syntaxe se nazývá Sass. Od verze 3 došlo ke změně syntaxe, která se nazývá Sassy CSS, zkráceně SCSS a je kompatibilní se syntaxí běžného CSS. Nejen Sass, ale preprocesory obecně přináší spoustu užitečných funkcí, které jazyk CSS neumožňuje a některé z nich si uvedeme.

***Nesting (Hnízdění)***

Je styl zápisu, který umožňuje zanořovat CSS pravidla do sebe a tím velice zpřehledňuje a zkracuje zápis. Je tu také možnost využívat klíčového znaku &, který značí rodičovský element.

Příklad takového zanoření může vypadat tímto způsobem:

```
div {
  background: black;
  color: white;
  p {
    color: red;
    &:hover {
      color: blue;
  }
}
```

Výše uvedený kód v jazyce CSS by vypadal takto:

```
div {
  background: black;
  color: white;
```

```
}
div p { color: red; }
div p:hover { color: blue; }
```

Jak je vidět výše, zápis pomocí SCSS syntaxe je přehlednější, protože všichni potomci elementu jsou v rámci jednoho bloku. Navíc Sass umožňuje vnořovat i media queries.

**Variables (Proměnné)**

Jsou klasické proměnné, jak je známe z programovacích jazyků. Řeší nám nepříjemný problém v kaskádových stylech, kdy např. při změně barvy je nutné projít veškerý kód a barvu přepsat. Pomocí proměnné si barvu deklarujeme a v CSS pravidlech místo konkrétní barvy zapíšeme proměnnou. V Sass se proměnná zapisuje klíčovým znakem $, po kterém následuje nazev-promenne, dvojtečka a hodnota.

```
$background: black;
$color-primary: white;
$color-secondary: red;

div {
  background: $background;
  color: $color-primary;
  p {
    color: $color-secondary;
    &:hover {
      color: $color-primary;
  }
}
```

**Mixins (Mixiny)**

Mixin je pojmenovaný kus kódu, který se opakuje a může mít vstupní parametry. Jedná se tedy o obdobu procedury v kaskádových stylech. Hodí se, pokud určitý kód chceme častěji používat. Příklad takového opakujícího kódu je vytvoření mixinu pro prefixy CSS vlastností. Mixin se definuje klíčovým slovem @mixin, po kterém následuje nazev-mixinu a závorky se vstupním parametrem.

```
@mixin transition-mixin($parametr) {
  -webkit-transition: $parametr;
  -moz-transition: $parametr;
  -o-transition: $parametr;
  transition: $parametr;
}
```

Následně se tento mixin vloží pomocí @include nazev-mixinu se zadaným parametrem.

```
.example {
  background: black;
  @include transition-mixin(all .3s);
}
.example2 {
  @include transition-mixin(all 1s);
}
```

**Extend (Rozšíření)**

Pomocí rozšíření je možné používat styly z již existujících selektorů. Jde tedy o dědičnost, která je běžná v objektově orientovaných jazycích. Volá se pomocí @extend nazev-selektoru.

```
.message {
  border: 1px solid green;
  color: yellow;
}
.warning {
  background: black;
  @extend .message;
}
.error {
  background: red;
  @extend .message;
}
```

Výsledný CSS kód bude poté vypadat následujícím způsobem:

```
.message, .warning, .error {
  border: 1px solid green;
  color: yellow;
}
.warning { background: black; }
.error { background: red; }
```

**Conditional Statements (Podmíněné příkazy)**

Sass také umožňuje využívat podmínky se standardní if/else konstrukcí. Příklad může vypadat tímto způsobem:

```
$var: true;

div {
  @if $var == true {
    background: red;
  }
  @else {
    background: black;
  }
}
```

Protože má proměnná $var definovanou hodnotu true, splní se podmínka a element div bude mít červené pozadí. Výsledný CSS kód poté vypadá takto:

```
div {
  background: red;
}
```

**Loops (Cykly)**

Další funkcí, kterou tento preprocesor nabízí, jsou cykly. Stejně jako programovací jazyky nabízí cyklus while, for a each. Praktický příklad each cyklu může být:

```
@each $icon in facebook, twitter, google-plus {
```

```
  .#{$icon}-icon {
    background-image: url("images/#{$icon}-icon.png");
 }
}
```

Ve výsledném CSS jsou poté tři třídy:

```
.facebook-icon {
  background-image: url("images/facebook-icon.png");
}
.twitter-icon {
  background-image: url("images/twitter-icon.png");
}
.google-plus-icon {
  background-image: url("images/google-plus-icon.png");
}
```

**Functions (Funkce)**

V tomto preprocesoru je možné definovat funkci, která bude vracet nějakou hodnotu. Definuje se klíčovým slovem @function nazev-funkce (vstupní parametry) a návratová hodnota je označená slovem @return.

```
@function moje-funkce($a, $b) {
  @return ($a + $b);
}
```

**Další možnosti**

Sass podporuje řadu dalších funkcí, kromě výše uvedených funkcí je možné provádět matematické výpočty, zaokrouhlovat čísla, vracet nejmenší nebo největší hodnotu z pole, importovat moduly a pracovat se seznamem dat. Umožňuje i práci s barvami, to znamená, že umí automaticky vrátit tmavší nebo světlejší odstín od zadané barvy, zvýšit průhlednost a mnoho dalších užitečných funkcí, které CSS neumožňují.

### 2.5.2 LESS

Tento preprocesor je dnes pravděpodobně druhým nejpoužívanějším a nejoblíbenějším frameworkem. Vznikl jako reakce na preprocesor Sass v roce 2009 (Spławski, 2015). Tedy v době, kdy Sass měl ještě původní syntaxi kódu, která více vyhovovala programátorům než kodérům. Less tedy vycházel ze syntaxe CSS, které doplnil o užitečné funkce. K jeho oblibě hodně pomohl CSS framework Bootstrap, který byl na tomto preprocesoru napsán. Ovšem v poslední verzi Bootstrapu již došlo k přepsání kódu do konkurenčního Sassu, což potvrzuje jeho aktuálně větší oblibu. V Lessu je možné zanořovat selektory do sebe, provádět matematické výpočty, vytvářet mixiny, pracovat s barvami, importovat moduly a další funkce.

Ukázka syntaxe jazyka:

```
@promenna: #c2c2c2;

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

.element {
  color: @promenna;
  .border-radius(5px);
  p {
    color: red;
    &:hover {
      color: blue;
    }
  }
}
```

Jak je vidět, syntaxe jazyka je hodně podobná preprocesoru Sass a zároveň se velice drží syntaxe kaskádových stylů. Takže i vytvoření např. mixinu vypadá jako vytvoření CSS třídy.

## 2.6 Automatizace

Při současném vývoji webových stránek se užívá mnoho technologií a rutinních postupů. Proto se využívá dalších užitečných nástrojů, které slouží k automatizování úkonů. Takovým automatizovaným procesem může být např. slučování CSS souborů do jednoho, doplňování prefixů CSS vlastnostem, minifikace JavaScript a CSS souborů, optimalizace obrázků, nebo třeba kompilace kódu z preprocesorů. Mezi nejznámější nástroje patří Gulp[7] a Grunt[8]. Oba si jsou velice podobné. Jsou napsány v JavaScriptu a pro svůj běh vyžadují serverový framework Node.js[9]. Ovládají se prostřednictvím příkazové řádky a jedná se o open source nástroje vydané pod licencí MIT (Ožana, 2014).

### 2.6.1 Grunt

Vznikl v roce 2012, jehož autorem je Ben Alman (Ožana, 2014). Grunt se označuje jako The Javascript Task Runner a jeho posláním je šetřit práci webovým vývojářům. Grunt tedy nedělá nic jiného, než že spouštění připravené úkoly, respektive pluginy. Hotových pluginů ať už přímo od vývojového týmu Gruntu, nebo od jiných vývojářů, existují tisíce. Aktuálně na oficiální webové stránce tohoto nástroje je možné stáhnout přes 6000 hotových pluginů[10]. Příkladem může být plugin: grunt-contrib-sass pro kompilaci Sass do CSS, nebo grunt-contrib-concat, který slučuje CSS soubory do jednoho. Pro práci s Gruntem jsou klíčové dva soubory: package.json a Gruntfile.

Package.json slouží k základní konfiguraci a k přidání pluginů, které Grunt bude používat. Použití tohoto souboru v jiném projektu umožňuje automatickou instalaci pluginů pomocí jednoho příkazu.

```
{
  "name": "Example",
  "version": "1.0.0",
  "description": "Příklad použití Gruntu",
  "devDependencies": {
    "grunt": "^1.0.1",
    "grunt-contrib-less": "*",
```

[7] Dostupné z: https://gulpjs.com/
[8] Dostupné z: https://gruntjs.com
[9] Dostupné z: https://nodejs.org/en/
[10] Dostupné z: https://gruntjs.com/plugins

```
    "grunt-contrib-watch": "*"
  }
}
```

Gruntfile.js slouží k samotné konfiguraci jednotlivých úkolů Gruntu.

```
module.exports = function(grunt) {
  grunt.initConfig({
    pkg: grunt.file.readJSON('package.json'),
    less: {
      development: {
        options: {
          compress: true,
        },
        files: {
          "css/style.css": "less/style.less"
        }
      }
    },

    watch: {
      styles: {
        files: ['less/**/*.less'],
        tasks: ['less']
      }
    }
  });
  grunt.registerTask('default', ['less', 'watch']);
}
```

Výše uvedený kód se stará o automatickou kompilaci z preprocesoru Less do CSS, jakmile dojde k nějaké změně v jakémkoli less souboru v projektu. Základem této automatizace je plugin grunt-contrib-watch, který kontroluje zadané soubory, a jakmile v nich nalezne změnu, volá úlohu less, která se stará o samotnou kompilaci.

### 2.6.2 Gulp

Vznikl v polovině roku 2013 (Ožana, 2014) a dělá obdobnou práci jako Grunt. Ovšem jde na to trochu jinou cestou a zaměřuje se více na psaní vlastního kódu oproti psaní konfigurace v Gruntu. Další rozdíl je, že využívá Node.js streamu, což znamená, že si mezivýsledky ukládá do paměti a úlohy provádí paralelně a tím i rychleji (Eckerstorfer, 2015).

Příklad automatické kompilace Less souborů do CSS pomocí Guplu vypadá tímto způsobem:

```
var gulp = require('gulp');
var less = require('gulp-less');

gulp.task('less', function() {
    return gulp.src('./style.less')
        .pipe(less())
        .pipe(gulp.dest('./build'))
});
gulp.task('watch', function() {
    gulp.watch('./*.less', ['less']);
});

gulp.task('default', ['watch']);
```

## 2.7 Shrnutí

Dnešní vývojáři a kodéři si pouze s kaskádovými styly příliš nevystačí. Je nutné si udržovat přehled o aktuálních trendech, postupech a snažit se je využívat. Výše uvedené metody, frameworky, preprocesory a snahy o automatizaci jsou dnes navzájem provázané úkony. CSS frameworky, které se dnes staly standardem při tvorbě frontendu webu jsou většinou postavené na nějakých objektových metodách, jako je např. OOCSS. Zároveň jsou tyto frameworky postavené na CSS preprocesorech, jejichž použití zase usnadňují nástroje jako je Gulp nebo Grunt. Výše uvedené technologie a metodiky jsou velmi efektivní při vývoji a umožňují snadné modifikace a postupné rozšiřování www aplikací.

# 3 Výuková prezentace

Jedním z cílů této bakalářské práce je vytvořit interaktivní výukovou prezentaci zaměřenou na aktuální CSS vlastnosti, která může sloužit jako výukový materiál pro žáky nebo začínající kodéry. Pod slovem aktuální si můžeme v podstatě představit všechny používané CSS vlastnosti od dob kaskádových stylů verze 1 z roku 1996 až po současnost. Ovšem aktuálním trendem jsou vlastnosti poslední verze CSS 3, případně vlastnosti, které jsou teprve ve vývoji. Proto v této prezentaci budou představeny některé z vlastností CSS 3 a pár vlastností, které již prošly čtvrtou iterací vývoje. Jelikož jsou kaskádové styly webovou technologií, vytvořená prezentace je ve formě jednoduché responzivní webové stránky.

Tato prezentace je umístěna na webovém serveru a je možné ji nalézt na URL adrese: http://bp.smarda.cz.

## 3.1 Použité technologie

Aby bylo možné takovou prezentaci vytvořit, je nutné si stanovit požadavky, které bude prezentace splňovat, a tomu přizpůsobit použité technologie.

Požadavky na tuto prezentaci tedy byly:

- responzivní webová stránka,
- podpora ve všech aktuálních prohlížečích,
- styl jako prezentace – slide přes celou obrazovku,
- možnost měnit parametry CSS vlastností,
- bez závislosti na frameworku,
- uplatnění metodologie BEM.

Prezentaci by bylo možné realizovat i pomocí nějakého frameworku, nebo CSS preprocesoru. Ovšem v tomto případě je to spíše nevhodné řešení. Kromě samotného použití HTML a kaskádových stylů je nutné využít i další technologie, jako je JavaScript.

Ve výukové prezentaci JavaScript slouží pro interaktivní změnu parametrů CSS vlastností uživatelem a pro vytvoření celostránkové prezentace. To znamená, že na celé ploše displeje se zobrazuje jeden snímek s jednou konkrétní CSS vlastností a ukázkou. Pohyb na další

snímek je možné učinit pomocí rolování myši, kurzorovými šipkami na klávesnici, nebo pomocí swipe pohybu na dotykových zařízeních.

K vytvoření této prezentace se využilo JavaScriptové knihovny fullPage[11] od autora Alvaro Trigo vydané pod licencí MIT. Tato knihovna existuje ve dvou verzích. První a stabilní verze závislá na JavaScriptovém frameworku jQuery[12]. Druhá verze je poměrně staršího data, neaktualizovaná, stále ve stavu alfa vývoje, ale bez závislosti na jQuery. Protože bylo v kritériích výše stanoveno, že nebude použit žádný framework, bude využito právě druhé verze této knihovny. Při měnění CSS vlastností v prezentaci by bylo vhodné použít nějakého posuvníku jako je input type="range" v HTML 5. Bohužel v knihovně fullPage se vyskytuje menší chyba, nebo spíše vlastnost, která na mobilních zařízeních reaguje na každý pohyb. Takže i na ten, který by se týkal pohybu posuvníku. Z tohoto důvodu je nutné v knihovně provést pár menších úprav, ale jinak se i vývojová verze jeví jako zcela funkční a bezchybná.

## 3.2 Vybrané CSS vlastnosti

Jak již bylo uvedeno na začátku této kapitoly, v prezentaci by se měly objevit některé vlastnosti z poslední verze kaskádových stylů. Nových vlastností je velké množství, ale ne všechny se jednoduše demonstrují ve formě prezentace. Proto se v prezentaci objeví ty nejzajímavější, nejužitečnější a zároveň dobře se prezentující vlastnosti. Detailnější popis těchto vlastností včetně jejich podpory v prohlížečích nalezneme níže.

**Columns**

První z nových CSS 3 vlastností, která bude v prezentaci, je vícesloupcové rozložení obsahu. To slouží k automatickému rozdělení textu do několika sloupců a vytváří se tak podobná sazba textu, jaká je použita v novinách.

CSS vlastnosti column jsou (W3Schools):

- ***column-count***
    - tímto parametrem se určuje počet sloupců,

[11] Dostupné z: https://alvarotrigo.com/fullPage/
[12] Dostupné z: http://jquery.com/

- ***column-width***
    - parametr této vlastnosti určuje šířku sloupce,
- ***column-gap***
    - slouží k nastavení šířky mezery mezi dvěma sloupci,
- ***column-rule***
    - umožňuje vytvořit a přizpůsobit oddělující čáru mezi sloupci.
    - Samotné column-rule je zkrácený zápis pro tyto vlastnosti:
        - ***column-rule-width***
            - tato vlastnost určuje tloušťku čáry,
        - ***column-rule-style***
            - touto vlastností se nastavuje styl čáry (jedná se o stejné parametry jako v případě vlastnosti border-style),
        - ***column-rule-color***
            - umožňuje nastavit barvu čáry,
- ***column-fill***
    - pomocí této vlastnosti je možné určit, jakým způsobem se budou sloupce plnit. Jsou dvě možnosti parametrů:
        - balance
            - při tomto parametru se budou sloupečky plnit rovnoměrně,
        - auto
            - s tímto parametrem se budou sloupečky plnit od prvního a budou tedy mít různou délku.

Ukázka použití při 3 sloupcích, které se automaticky roztáhnout na maximální šířku rodičovského elementu s nastavenou dělicí čárou, může vypadat třeba takto:

```
.element {
  column-count: 3;
  column-rule: 2px solid red;
  column-gap: 10px;
  column-fill: balance;
```

```
}
```

*Tabulka 3 Podpora sloupců v prohlížečích*

| | IE | Chrome | Firefox | Opera | Safari |
|---|---|---|---|---|---|
| Podpora od verze | 10.0 | 50.0 | 52.0 | 37 | 9.0 |
| Podpora s prefixem od verze | | 4.0 | 2.0 | 11.1 | 3.1 |

Zdroj: CSS Multiple Columns. *W3 schools* [online]. [cit. 2018-04-05]. Dostupné z: https://www.w3schools.com/css/css3_multiple_columns.asp

**Shadows**

Další ukázkou v prezentaci a zároveň užitečnou funkcí v CSS 3 jsou stíny. Stín je možné aplikovat buď na text (text-shadow), nebo na nějaký element (box-shadow). Obě vlastnosti se chovají a zapisují stejným způsobem, až na jednu výjimku a tou je klíčové slovo inset, které není možné použít na stín textu.

Stín u textu ovšem není tak úplně novinkou, objevil se již ve specifikaci CSS 2 (Dudek, 2002). Ale větší oblibu si získal v souvislosti s třetí verzí kaskádových stylů, W3Schools dokonce text-shadow řadí do CSS 3 a Internet Explorer podporoval dříve vlastnost box-shadow (W3Schools).

Parametry obou funkcí v jejich pořadí zápisu:

- horizontální posun stínu,
- vertikální posun stínu,
- poloměr rozostření,
- poloměr šíření stínu,
- volitelná vlastnost inset, kterou je možné aplikovat pouze na stín v boxu. Mění vržený vnější stín na vnitřní.

Ukázka použití:

```
.element {
  box-shadow: 10px 10px 10px rgba(0,0,0,0.5) inset;
  text-shadow: 5px 5px 5px rgba #bbb;
}
```

*Tabulka 4 Podpora vlastnosti text-shadow v prohlížečích*

| | IE | Chrome | Firefox | Opera | Safari |
|---|---|---|---|---|---|
| Podpora od verze | 10.0 | 4.0 | 3.5 | 9.6 | 4.0 |

Zdroj: CSS text-shadow Property. *W3 schools* [online]. [cit. 2018-04-05]. Dostupné z: https://www.w3schools.com/css/css3_pr_text-shadow.asp

*Tabulka 5 Podpora vlastnosti box-shadow v prohlížečích*

| | IE | Chrome | Firefox | Opera | Safari |
|---|---|---|---|---|---|
| Podpora od verze | 9.0 | 10.0 | 4.0 | 10.5 | 5.1 |
| Podpora s prefixem od verze | | 4.0 | 3.5 | 11.1 | 3.1 |

Zdroj: CSS box-shadow Property. *W3 schools* [online]. [cit. 2018-04-05]. Dostupné z: https://www.w3schools.com/css/css3_pr_box-shadow.asp

**Border radius**

Border radius (zakulacené rohy) řeší poměrně nepříjemnou situaci. Nyní už není nutné pro kulaté rohy u elementů využívat průhledných obrázků, ale jednoduše se zaobli pomocí CSS. Dokonce je možné zakulatit každý roh zvlášť a jiným poloměrem. Pomocí této vlastnosti je možné vytvořit i kruh.

Tuto CSS vlastnost je možné zapsat pro každý roh zvlášť, nebo jako zkrácený zápis.

```
.element {
  border-top-left-radius: 10px;
  border-top-right-radius: 10px;
  border-bottom-right-radius: 10px;
  border-bottom-left-radius: 10px;
```

```
  /* ekvivalent zápisu */
  border-radius: 10px 10px 10px 10px;
}
```

*Tabulka 6 Podpora vlastnosti border-radius v prohlížečích*

| | IE | Chrome | Firefox | Opera | Safari |
|---|---|---|---|---|---|
| Podpora od verze | 9.0 | 5.0 | 4.0 | 10.5 | 5.0 |
| Podpora s prefixem od verze | | 4.0 | 3.0 | 11.1 | 3.1 |

Zdroj: CSS border-radius Property. *W3 schools* [online]. [cit. 2018-04-05]. Dostupné z: https://www.w3schools.com/css/css3_pr_border-radius.asp

**Opacity**

Kaskádové styly umožňují využít při zápisu barvy v RGB, nebo v HSL modelu alfa kanálu. Tím je možné nastavit průhlednost pozadí nebo písma. Pokud ovšem chceme celý element nastavit průhledný, nebo nastavit průhlednost obrázku, slouží k tomu právě vlastnost opacity. Ta může nabýt hodnoty 0 až 1. Při nule je element absolutně průhledný a při jedničce je zcela neprůhledný.

```
.element {
  opacity: 0.7;
}
```

*Tabulka 7 Podpora vlastnosti opacity v prohlížečích*

| | IE | Chrome | Firefox | Opera | Safari |
|---|---|---|---|---|---|
| Podpora od verze | 9.0 | 4.0 | 2.0 | 9.0 | 3.1 |
| Podpora s prefixem od verze | Pro starší verze je možné použít filter: alpha(opacity=X) | | | | |

Zdroj: CSS opacity Property. *W3 schools* [online]. [cit. 2018-04-05]. Dostupné z: https://www.w3schools.com/css/css3_pr_opacity.asp

**Transform**

Pomocí této vlastnosti je možné různě transformovat elementy. Například je otáčet, posouvat, zvětšovat ve dvou nebo i třech rozměrech. Kvůli jednoduché demonstraci budou v prezentaci zobrazeny pouze 2D transformace.

Parametry transform vlastnosti jsou (W3Schools):

- ***translate(x, y)***
  - Slouží k posunutí elementu o určitou vzdálenost v ose X, Y i ve 3D ose Z.
  - translate3d(x, y, z), translateX(x), translateY(y), translateZ(z)
- ***scale(x, y)***
  - Tento parametr slouží ke zvětšení elementu, výchozí hodnota je 1, tzn. hodnota, při které je element v původní velikosti. Zvětšení je možné provést v ose X, Y i Z.
  - scale3d(x, y, z), scaleX(x)**,** scaleY(y)**,** scaleZ(z)
- ***rotate(stupně)***
  - Jak již z názvu vyplývá, tímto parametrem je možné otáčet elementy. Hodnotou jsou stupně a je opět možné pracovat v ose X, Y a Z.
  - rotate(x, y, z, stupně), rotateX(stupně), rotateY(stupně), rotateZ(stupně)
- ***skew(x-stupně, y-stupně)***
  - Tímto parametrem je možné provést zkosení elementu v ose X a Y. Hodnotou parametru jsou opět stupně.
  - skewX(stupně), skewY(stupně)
- ***matrix(n, n, n, n, n, n)***
  - Tímto parametrem je možné kombinovat všechny transformační metody do jedné jako matematické funkce.
- ***matrix3d(n, n, n, n, n, n, n, n, n, n, n, n, n, n, n, n)***
  - Je obdobným parametrem ovšem i pro 3D transformaci.
- ***perspective(n)***
  - Tento parametr slouží ke zkreslení perspektivy ve 3D prostoru.

Příklad otočení elementu:

```
.element {
  transform: rotate(180deg);
}
```

*Tabulka 8 Podpora vlastnosti transform v prohlížečích*

| | IE | Chrome | Firefox | Opera | Safari |
|---|---|---|---|---|---|
| Podpora od verze (2D) | 10.0 | 36.0 | 2.0 | 9.0 | 9.0 |
| Podpora s prefixem od verze (2D) | 9.0 | 4.0 | 3.5 | 10.5 | 3.2 |
| Podpora od verze (3D) | 12 | 36.0 | 10.0 | 23.0 | 9.0 |
| Podpora s prefixem od verze (3D) | | 12.0 | | 15.0 | 4.0 |

Zdroj: CSS transform Property. *W3 schools* [online]. [cit. 2018-04-05]. Dostupné z: https://www.w3schools.com/css/css3_pr_transform.asp

**Perspective**

Tato vlastnost má obdobnou funkci jako transform: perspective jen s tím rozdílem, že se tato vlastnost neaplikuje na element samotný, ale na jeho potomky. To znamená, že je nutné vytvořit rodičovský element s touto vlastností a v něm potomky, na kterých bude aplikována libovolná vlastnost transform. Poté se na potomcích aplikuje jak vlastnost transform, tak vlastnost perspective. Kromě samotné vlastnosti perspective ještě existuje perspective-origin, která určuje, v jaké poloze se uživatel na 3D prvek dívá.

Ukázka použití této vlastnosti:

```
<div class="parent">
  <div class="child"></div>
```

```
</div>
.parent {
  perspective: 500px;
  perspective-origin: 20% 50%;
}
.child {
  transform: rotate(45deg);
}
```

*Tabulka 9 Podpora vlastnosti perspective v prohlížečích*

| | IE | Chrome | Firefox | Opera | Safari |
|---|---|---|---|---|---|
| Podpora od verze | 10.0 | 36.0 | 16.0 | 23.0 | 9.0 |
| Podpora s prefixem od verze | | 12.0 | 10.0 | 15.0 | 4.0.3 |

Zdroj: CSS perspective Property. *W3 schools* [online]. [cit. 2018-04-05]. Dostupné z: https://www.w3schools.com/css/css3_pr_perspective.asp

**Filter**

S prvními filtry přišly už staré verze Internet Exploreru, které se současnou implementací v prohlížečích nemají nic společného (Jahoda, 2015). Tato vlastnost slouží k vytvoření různých efektů, jako je rozmazání, zvýšení kontrastu nebo inverze barev. Efekt je možné aplikovat na obrázky i na většinu elementů.

Parametry této vlastnosti jsou následující:

- ***blur***
  - Pomocí tohoto parametru lze aplikovat efekt rozmazání.
- ***brightness(%)***
  - Tímto parametrem můžeme ztmavit, nebo naopak projasnit element. Hodnotou je procento, kde 0% je minimální jas, tzn. element je černý, a naopak maximální hodnotou je 200%, kdy je element úplně bílý.

- ***contrast(%)***
    - Efekt, který slouží ke změně kontrastu. Hodnotou jsou opět procenta, kde 0% nastaví nulový kontrast a 200% nastaví maximální kontrast.
- ***drop-shadow***
    - Jedná se o obdobnou funkci jako je box-shadow i se stejnými parametry.
- ***grayscale(%)***
    - Dalším poměrně užitečným efektem je grayscale, který umožňuje z barevného elementu udělat černobílý. Hodnota 100% nastaví element s maximálním černobílým efektem.
- ***hue-rotate(stupně)***
    - Tímto efektem můžeme posunout barvy do jiného spektra, a tím dosáhnout jiné barevnosti všech elementů, na které je tento filtr aplikován.
- ***invert(%)***
    - Jak už plyne z názvu, jedná se o efekt, který invertuje barvy.
- ***saturate(%)***
    - Po aplikaci tohoto filtru je možné ovlivňovat sytost barev. Hodnotou jsou opět procenta od 0 do 200.
- ***opacity(%)***
    - Tento filtr je shodný se samostatnou CSS vlastností opacity a umožňuje tedy vytvořit efekt průhlednosti. Jediný rozdíl je, že zde se zadává hodnota v procentech.
- ***sepia(%)***
    - Filtr sépia slouží k navození efektu staré fotografie od 0 do 100%.

Příklad použití:

```
img {
  filter: blur(15px);
}
```

*Tabulka 10 Podpora CSS vlastnosti filter v prohlížečích*

| | IE | Chrome | Firefox | Opera | Safari |
|---|---|---|---|---|---|
| Podpora od verze | 13.0 | 53.0 | 35.0 | 40.0 | 9.1 |
| Podpora s prefixem od verze | | 18.0 | | 15.0 | 6.0 |

Zdroj: CSS filter Property. *W3 schools* [online]. [cit. 2018-04-05]. Dostupné z: https://www.w3schools.com/css/css3_pr_filter.asp

**Gradients**

Gradienty jsou další novinkou, které umožňují vytvořit barevné přechody přímo pomocí kaskádových stylů. Přechody se zapisují jako parametr CSS vlastnosti background, nebo background-image.

Existují 3 druhy přechodů:

- Linear Gradients (Lineární gradient)
    - Je základním typem přechodu, kterému je možné měnit směr vykreslení. Základní směry jsou zdola nahuru (to top), shora dolů (to bottom), zleva doprava (to right), a zprava doleva (to left). Kromě těchto základních směrů lze použít i šikmé směry jako např. to bottom right, nebo určit směr pomocí stupňů.
- Radial gradient (Radiální gradient)
    - Dalším typem přechodu je kruhový gradient, který má dva parametry. Tyto parametry udávají finální tvar přechodu. Výchozím tvarem je ellipse, ten roztáhne přechod přes celý element. Druhým tvarem je circle, který vytvoří ve středu přechodu kruh.
- Repeating gradient (Opakující se gradient)
    - Tento typ provádí nekonečné opakování lineárního i radiálního přechodu.

Příklad jednoduchého přechodu:

```
.element {
```

```
  background-image: linear-gradient(to right, red, blue);
}
```

*Tabulka 11 Podpora přechodů v prohlížečích*

| | IE | Chrome | Firefox | Opera | Safari |
|---|---|---|---|---|---|
| Podpora od verze | 10.0 | 26.0 | 16.0 | 12.1 | 6.1 |
| Podpora s prefixem od verze | | 10.0 | 3.6 | | 5.1 |

Zdroj: Can I use linear-gradient. *Can I use* [online]. [cit. 2018-04-06]. Dostupné z: https://caniuse.com/#search=radial-gradient

**Transitions**

Další ukázkou v pořadí jsou CSS přechody. Tato vlastnost umožňuje animovat stavy mezi přechody elementů. Běžně se k tomu v kaskádových stylech využívá pseudotřídy :hover, která mění stav elementu po najetí kurzorem. Příkladem může být změna barvy písma nebo šířky. Jakmile se tedy elementu přidá vlastnost transition, je možné docílit jejich plynulého prolnutí z původní hodnoty na novou.

Transition vlastnosti jsou následující (Michálek, 2017):

- ***transition-property***
  - Transition umožňuje animovat všechny vlastnosti elementu, nebo jen některé. K tomu právě slouží tato vlastnost, jejíž hodnotou je klíčové slovo all, které říká, že se mají animovat všechny vlastnosti. Pokud ovšem chceme animovat pouze nějakou konkrétní vlastnost např. barvu písma, místo all uvedeme název vlastnosti, tedy color.
- ***transition-duration***
  - Je povinná vlastnost, která určuje délku animace. Hodnotu je možné zadávat v sekundách nebo milisekundách.

- ***transition-timing-function***
    - Slouží k ovlivnění průběhu animace. Tzn. že je možné animovat pomalu s postupným zrychlením, nebo třeba naopak. CSS má pět přednastavených stylů animace a umožňuje si vytvořit i vlastní typ animace.
    - Parametrem této vlastnosti tedy může být:
        - linear – stejná rychlost animace od začátku až do konce,
        - ease – pomalý start, ve středu animace dojde ke zrychlení a ke konci se opět zpomaluje,
        - ease-in – pomalejší start animace,
        - ease-in-out – tento typ animace má pomalejší start i konec,
        - cubic-bezier – umožňuje definovat vlastní efekt.
- ***transition-delay***
    - Touto vlastností lze oddálit spuštění animace. Čas se zadává opět v sekundách, nebo milisekundách.

Příklad jednoduchého přechodu všech vlastností:

```
.element {
  transition-property: all;
  transition-timing-function: ease;
  transition-duration: 2s;
  transition-delay: 1s;
  /* zkrácený zápis */
  transition: all 2s ease 1s;
}
```

*Tabulka 12 Podpora transition v prohlížečích*

| | IE | Chrome | Firefox | Opera | Safari |
|---|---|---|---|---|---|
| Podpora od verze | 10.0 | 26.0 | 16.0 | 12.1 | 6.1 |
| Podpora s prefixem od verze | | 4.0 | 4.0 | 10.5 | 3.1 |

Zdroj: CSS Transitions. *W3 schools* [online]. [cit. 2018-04-05]. Dostupné z: https://www.w3schools.com/css/css3_transitions.asp

**Animations**

Kromě výše zmíněných transition je Animations dalším typem animací. Tato vlastnost je přímo nativní podporou pro složitější a plnohodnotnější animace v CSS (Jahoda, 2013). Již tedy není nutné používat animované obrázky, flash, nebo jiné technologie jako je JavaScript.

Jelikož se složitější ukázka poměrně těžce demonstruje, je v prezentaci pouze připravená interaktivní animace, kterou je možné přizpůsobovat změnou parametrů, a tím ukázat možnosti kaskádových stylů.

Aby bylo možné animace používat, je nejprve nutné nadefinovat pomocí klíčového slova @keyframes samotnou animaci. Zde se nastaví začátek, průběh a konec animace. Takto vytvořenou animaci je následně možné odkudkoli zavolat a přizpůsobit požadovaným vlastnostem.

CSS animation má hodně vlastností, díky kterým je možné každou animaci přizpůsobit (Michálek, 2017).

- ***animation-name***
    - Tato vlastnost má jako parametr název volané animace.
- ***animation-duration***
    - Určuje délku trvání animace v sekundách nebo milisekundách.
- ***animation-timing-function***
    - Obdobně jako u transition slouží tato vlastnost k ovlivnění průběhu animace včetně stejných hodnot, které může nabývat.
- ***animation-delay***
    - Touto vlastností je možné nastavit zpoždění, za které se animace spustí.
- ***animation-iteration-count***
    - Animace je možné spouštět stále dokola, nebo nastavit přesný počet opakování. Toho lze docílit pomocí této vlastnosti zadáním slova infinite, nebo čísla určujícího počet opakování.

- ***animation-direction***
    - Určuje směr průběhu animace a může nabývat těchto parametrů:
        - normal – animace je spuštěna od 0% do 100% a v případě opakování skočí na 0%,
        - reverse – animace běží od 100% k 0%, v případě opakování skočí na 100%,
        - alternate – animace se spustí od 0% do 100%, v případě opakování bude animace pokračovat od 100% k 0% a tak stále dokola,
        - alternate-reverse – tento parametr se chová jako předešlý jen v opačném směru.
- ***animation-fill-mode***
    - Určuje, zda první, nebo poslední snímek má nabývat hodnoty z keyframe před, nebo po spuštění animace. Možné parametry jsou tyto:
        - none – před animací se nic nezmění, po animace se vše vrátí na původní hodnoty elementu,
        - forwards – po skončení animace zůstanou elementu vlastnosti z posledního snímku (keyframe 100%),
        - backwards – pokud se nastaví animation-delay, první snímek převezme vlastnosti z keyframe 0%
        - both – aplikuje se forwards i backwards.
- ***animation-play-state***
    - Umožňuje pozastavit a opětovně spouštět animaci. Toho je možné dosáhnout pomocí parametru paused a running.

Příklad animace:

```
@keyframes nazev_animace {
  0%   { background-color: red; top: 0; }
  25%  { background-color: yellow; top: 200px; }
  100% { background-color: blue; top: 0; }
}
```

```
.element {
  background-color: black;
  animation-name: nazev_animace;
  animation-duration: 3s;
  animation-delay: 2s;
  animation-fill-mode: backwards;
}
```

*Tabulka 13 Podpora animací v prohlížečích*

| | IE | Chrome | Firefox | Opera | Safari |
|---|---|---|---|---|---|
| Podpora od verze | 10.0 | 43.0 | 16.0 | 30.0 | 9.0 |
| Podpora s prefixem od verze | | 4.0 | 5.0 | 12.0 | 4.0 |

Zdroj: CSS Animations. *W3 schools* [online]. [cit. 2018-04-05]. Dostupné z: https://www.w3schools.com/css/css3_animations.asp

**Selectory**

CSS 3 přinesly spoustu nových selektorů jako je :first-of-type, :last-of-type a řadu dalších. V prezentaci by jejich demonstrace nebyla příliš zajímavá, a proto se zaměříme jen na nové. Pod pojem nové selektory je možné zařadit ty, které spadají pod „CSS 4“. Respektive jsou to ty, které prošly čtvrtou iterací vývoje. Tyto selektory jsou stále ve vývoji a jejich podpora v prohlížečích je aktuálně velice špatná a dosti rozdílná.

Protože není jisté, které selektory a kdy se rozhodnou výrobci prohlížečů implementovat, je obtížné zařadit je do prezentace. Z tohoto důvodu se v prezentaci objeví 12 nových selektorů, které se dají dobře demonstrovat a všechny jsou podporovány minimálně v jednom prohlížeči.

Selektory, které splňují tyto podmínky a je možné je zařadit do prezentace, jsou uvedeny níže.

***Case Sensitivity***

Selektor, který umožňuje vybrat element bez ohledu na velikost písmen. Příklad takového selektoru může být výběr takových odkazů, které obsahují v parametru href slovo pdf bez ohledu, zda jsou zapsány jako pdf, PDF, nebo pDf (Selectors Level 4, 2018).

```
a[href$="pdf" i] {
  color: red;
}
```

***Placeholder***

Placeholder je již poměrně delší dobu dobře podporovaný selektor, který slouží ke změně ukázkové hodnoty v input elementu. Například změnu barvy písma na červenou je možné udělat tímto způsobem:

```
:placeholder-shown { color: red; }
::-webkit-input-placeholder { color: red; }
::-moz-placeholder { color: red; }
:-ms-input-placeholder { color: red; }
```

***Optionality [:required] a [:optional]***

Dalším novým selektorem je :required, který vybírá ty input elementy, které mají nastavený parametr required. A naopak :optional, který vybírá inputy, které tento parametr neobsahují.

```
:required { border: 2px solid red; }
:optional { border: 2px solid green; }

<input type="text" required="required">
<input type="text"> /* tento input bude se zeleným okrajem
*/
```

***Mutability [:read-write] a [:read-only]***

Tento selektor slouží pro výběr jen těch elementů, ze kterých je možné číst i do nich zapisovat, nebo pouze číst.

```
:read-write { border: 2px solid red; }
:read-only { border: 2px solid green; }

<input type="text"> /* vyhoví selektoru :read-write */
<p contenteditable="true"> /* taktéž :read-write */

<input type="text" readonly> /* vyhoví selektoru :read */
```

***Range [:valid] a [:invalid]***

Pomocí tohoto selektoru je možné vybrat a stylovat ty elementy, u kterých obsah odpovídá, nebo naopak neodpovídá typu inputu.

```
:valid { border: 2px solid green; }
:invalid { border: 2px solid red; }

<input type="email" value="uzivatel@email.cz"> /* zadaná
hodnota odpovídá typu */

<input type="email" value="@email.cz"> /* zadaná hodnota
neodpovídá typu a bude tedy mít červený okraj */
```

***Validity [:in-range] a [:out-of-range]***

Slouží pro výběr těch elementů, které mají hodnotu v zadaném rozsahu.

```
:in-range { border: 2px solid green; }
:out-of-range { border: 2px solid red; }

<input type="number" min="1" max="5" value="3"> /* hodnota
3 je v rozsahu */

<input type="number" min="1" max="5" value="7"> /* hodnota
7 není v rozsahu a bude s červeným okrajem */
```

***Indeterminate***

Poměrně zajímavým selektorem je indeterminate, který slouží ke stylování neurčitého stavu u checkboxu. To znamená, že se jedná o třetí stav checkboxu, nebo radio inputu mimo jeho běžné stavy, kde je, nebo není zaškrtnut. Aktuálně je možné na tento neurčitý stav aplikovat pouze průhlednost.

```
:indeterminate { opacity: 0.5; }
<input type="checkbox">
```

***Default***

Slouží pro výběr výchozích elementů. To je možné aplikovat např. na řadu radio inputů.

```
:default + label { color: red; }

<input type="radio" id="a"> <label for="a">A</label>

<input type="radio" id="b" checked> <label
for="b">B</label> /* Tento input má parametr checked, takže
se jedná o výchozí element */
```

## 3.3 Podpora v prohlížečích

Prezentace se zaměřuje na ukázku aktuálních možností kaskádových stylů, a proto je podpora v prohlížečích zaměřená primárně na všechny běžně používané prohlížeče současnosti. Samozřejmě s velkým důrazem na podporu v mobilních zařízeních. Kaskádové styly ve třetí verzi jsou dnes již dobře podporovány, což je i uvedeno v tabulkách u každé výše uvedené vlastnosti. Na zařízení s aktualizovaným webovým prohlížečem není použití výukové prezentace a obecně CSS 3 žádný problém. S ohledem na užité technologie je nutné použít minimálně Internet Explorer ve verzi 10. Ve starších verzích nebude prezentace zobrazena správně a nebude možné využít všech demonstrovaných funkcí.

Ovšem podpora nových CSS vlastností jako uvedené selektory úrovně 4 není zaručena ani v nejnovějších prohlížečích. Aktuální test podpory v nejnovějších verzích prohlížečů v systému Windows 10 vypadá následně:

*Tabulka 14 Podpora nových selektorů v prohlížečích*

| | IE 9 | IE 10 | IE 11 | MS Edge 41.16 | Firefox 59 | Chrome 65 |
|---|---|---|---|---|---|---|
| Case sensitivity | × | × | × | × | ✓ | ✓ |
| Placeholder | × | ✓ | ✓ | ✓ | ✓ | ✓ |
| Optionality | × | ✓ | ✓ | ✓ | ✓ | ✓ |
| Mutability | × | × | × | ✓ | × | ✓ |
| Range | × | ✓ | ✓ | ✓ | ✓ | ✓ |
| Validity | × | × | × | ✓ | ✓ | ✓ |
| Indeterminate | × | ✓ | ✓ | ✓ | ✓ | ✓ |
| Default | × | × | × | × | ✓ | ✓ |

Do prezentace byly záměrně vybrány selektory, které budou alespoň v jednom prohlížeči zcela funkční. Tímto prohlížečem je Chrome, potažmo prohlížeče postavené na stejném jádře jako je i např. Opera. Existuje ale i spousta dalších selektorů, které nejsou implementovány v žádném aktuálním prohlížeči. Doporučeným prohlížečem pro zobrazení prezentace, ve kterém budou fungovat všechny demonstrované ukázky, je tedy Chrome v co nejaktuálnější verzi.

# Závěr

Teoretická část této bakalářské práce přibližuje historický vývoj kaskádových stylů, podává přehled o aktuálních trendech a možnostech při stylování webových stránek. Je zde mapován vývoj kaskádových stylů od jejich vzniku přes současnost až po náhled do blízké budoucnosti. Práce je věnována aktuálním trendům, jako jsou metodiky a objektové psaní kaskádových stylů, CSS frameworkům, preprocesorům i automatizačním nástrojům.

Jsou zde uvedeny a vysvětleny konkrétní ukázky použití metodik, jako je OOCSS, SMACSS, nebo BEM, příklady použití frameworku Bootstrap, preprocesoru SASS a nástroje Grunt.

Praktická část je zaměřena na demonstraci aktuálních vlastností kaskádových stylů pomocí interaktivní webové prezentace. Představeny jsou vlastnosti jako stíny, přechody, filtry nebo nové selektory. V práci je také uvedeno, jak jsou tyto vlastnosti podporovány v běžných webových prohlížečích. Díky své interaktivitě může tato prezentace sloužit jako výukový materiál pro studenty i začínající kodéry. Prezentace je umístěna na webovém serveru a je možné ji nalézt na URL adrese: http://bp.smarda.cz.

Lze tedy říci, že dnes si vývojáři webových stránek pouze s kaskádovými styly příliš nevystačí. Je nutné mít přehled o všech aktuálních technologiích, metodách a postupech vývoje. To je ovšem poměrně náročné, jelikož se jedná o velice komplexní problematiku, která podléhá rychlému vývoji.

## Seznam použitých informačních zdrojů


1) 13 trendů webdesignu pro rok 2017. *MoAdvertising* [online]. 24. 3. 2017 [cit. 2018-03-21]. Dostupné z: http://moadvertising.cz/archives/650
2) A brief history of CSS until 2016. *Writings of a Page Load Speed Geek* [online]. 11. 4. 2016 [cit. 2018-03-15]. Dostupné z: http://www.css-class.com/a-brief-history-of-css/
3) A Yeti's Trek. *Foundation* [online]. [cit. 2018-04-01]. Dostupné z: https://foundation.zurb.com/showcase/about.html
4) About. *Bootstrap* [online]. [cit. 2018-03-30]. Dostupné z: https://getbootstrap.com/docs/3.3/about/
5) BERNER, David. Battling BEM CSS: 10 Common Problems And How To Avoid Them. *Smashing magazine* [online]. 1. 6. 2016 [cit. 2018-03-28]. Dostupné z: https://www.smashingmagazine.com/2016/06/battling-bem-extended-edition-common-problems-and-how-to-avoid-them/
6) BOS, Bert. A brief history of CSS until 2016. *W3* [online]. 17. 12. 2016 [cit. 2018-03-15]. Dostupné z: http://www.w3.org/Style/CSS20/history.html
7) BRDA, Jiří. 9 nejnovějších webdesignových trendů pro rok 2017. *Jiří Brda graphic designer* [online]. 9. 1. 2017 [cit. 2018-03-21]. Dostupné z: http://www.jiribrda.cz/9-nejnovejsich-webdesignovych-trendu-pro-rok-2017.html
8) CYROŇ, Miroslav. *CSS - kaskádové styly: praktický manuál*. Praha: Grada, 2006. Průvodce (Grada). ISBN 80-247-1420-5.
9) ČÁPKA, David. 26. díl - Bootstrap - Úvod do grid systémů. *ITnetwork* [online]. [cit. 2018-03-31]. Dostupné z: https://www.itnetwork.cz/html-css/bootstrap/bootstrap-uvod-do-grid-systemu
10) CSS Tutorial. *W3 schools* [online]. [cit. 2018-04-05]. Dostupné z: https://www.w3schools.com/css
11) DUDEK, Jan. CSS2 – kaskádové styly podruhé. *Interval* [online]. 4. 12. 2012 [cit. 2018-04-06]. Dostupné z: https://www.interval.cz/clanky/css2-kaskadove-styly-podruhe/

12) ECKERSTORFER, Florian. *Florian Eckerstorfer* [online]. 24. 6. 2015 [cit. 2018-04-03]. Dostupné z: https://florian.ec/articles/gulp-js-streams/
13) GASSTON, Peter. *CSS3*. Přeložil Ondřej BAŠE. Brno: Computer Press, 2016. ISBN 978-80-251-4641-5.
14) GASSTON, Peter. *Moderní web.* Přeložil Ondřej BAŠE. Brno: Computer Press, 2015. ISBN 978-80-251-4345-2.
15) Framework. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2017 [cit. 2018-03-29]. Dostupné z: https://cs.wikipedia.org/wiki/Framework
16) HAUSER, Marianne, Tobias HAUSER a Christian WENZ. *HTML a CSS: velká kniha řešení*. Brno: Computer Press, 2006. ISBN 80-251-1117-2.
17) HTML 5.2. *W3C* [online]. 2017 [cit. 2018-04-05]. Dostupné z: https://www.w3.org/TR/2017/REC-html52-20171214/
18) ILLY, Patrik, FRONTENDISTI, ed. OOCSS / SMACSS / BEM (Organizace kódu) - Frontendisti.cz. Youtube [online]. 4. 5. 2014 [cit. 2018-03-24]. Dostupné z: https://www.youtube.com/watch?v=obx1PawM2WE
19) JAHODA, Bohumil. CSS filter. *Je čas* [online]. 21. 1. 2015 [cit. 2018-04-08]. Dostupné z: http://jecas.cz/filter
20) JAHODA, Bohumil. CSS vlastnost animation *Je čas* [online]. 25. 9. 2013 [cit. 2018-04-08]. Dostupné z: http://jecas.cz/animation
21) JANOVSKÝ, Dušan. Historie CSS. *Jak psát web* [online]. [cit. 2018-03-15]. Dostupné z: https://www.jakpsatweb.cz/css/css-historie.html
22) JAVOREK, Honza. CSS preprocesory: méně psaní, vyšší efektivita. *Zdroják* [online]. 29. 4. 2011 [cit. 2018-04-01]. Dostupné z: https://www.zdrojak.cz/clanky/css-preprocesory-mene-psani-vyssi-efektivita/
23) Kaskádové styly. *Adaptic* [online]. [cit. 2018-03-15]. Dostupné z: http://www.adaptic.cz/znalosti/slovnicek/kaskadove-styly/
24) KOSEK, Jiří. Historie a vývoj HTML. *HTML guru* [online]. [cit. 2018-04-05]. Dostupné z: http://htmlguru.cz/uvod-historie.html
25) KOVAŘÍK, Pavel. Co bude udávat trendy při tvorbě webových stránek v roce 2018?. *Kovařík media* [online]. 2018 [cit. 2018-03-21]. Dostupné z:

http://kovarikmedia.cz/clanek/co-bude-udavat-trendy-pri-tvorbe-webovych-stranek-v-roce-2018

26) KOZLEROVÁ, Jana. Parallax. *CSS blog* [online]. 8. 4. 2016 [cit. 2018-03-21]. Dostupné z: http://cssblog.antee.cz/blog/parallax

27) Media Queries. *Foundation for Sites 6 Docs* [online]. [cit. 2018-04-01]. Dostupné z: https://foundation.zurb.com/sites/docs/media-queries.html

28) MICHÁLEK, Martin. BEM: Pojmenovávací konvence pro třídy v CSS. *Vzhůru dolů* [online]. 1. 9. 2017 [cit. 2018-03-28]. Dostupné z: https://www.vzhurudolu.cz/prirucka/bem

29) MICHÁLEK, Martin. *Vzhůru do CSS3* [epub]. Michálek Martin - Vzhůru dolů, 2017 [cit. 2018-03-24]. ISBN 978-80-260-8438-9. Dostupné z: https://www.vzhurudolu.cz/ebook-css3

30) NEUMANN, Michal. Web design trendy 2015. *Litea Solution* [online]. 18. 12. 2014 [cit. 2018-03-21]. Dostupné z: https://www.litea.cz/clanky/detail/web-design-trendy-2015

31) OTTO, Mark. Bootstrap 4. The Bootstrap Blog [online]. 18. 1 .2018 [cit. 2018-03-30]. Dostupné z: https://blog.getbootstrap.com/2018/01/18/bootstrap-4/

32) OTTO, Mark. Say hello to Bootstrap 2.0. *Developer Blog* [online]. 31. 1. 2012 [cit. 2018-03-30]. Dostupné z: https://blog.twitter.com/developer/en_us/a/2012/say-hello-to-bootstrap-2.html

33) OTTO, Mark. *The Bootstrap Blog* [online]. 29. 11. 2014 [cit. 2018-03-30]. Dostupné z: https://blog.getbootstrap.com/2014/10/29/bootstrap-3-3-0-released/

34) OŽANA, Roman. *Zdroják* [online]. 22. 8. 2014 [cit. 2018-04-03]. Dostupné z: https://www.zdrojak.cz/clanky/gulp-vs-grunt-souboj-bez-viteze-a-porazeneho/

35) PÍSEK, Slavoj. *HTML: začínáme programovat.* 3., aktualiz. vyd. [i.e.] 1. vyd. Praha: Grada, 2010. ISBN 8024731177.

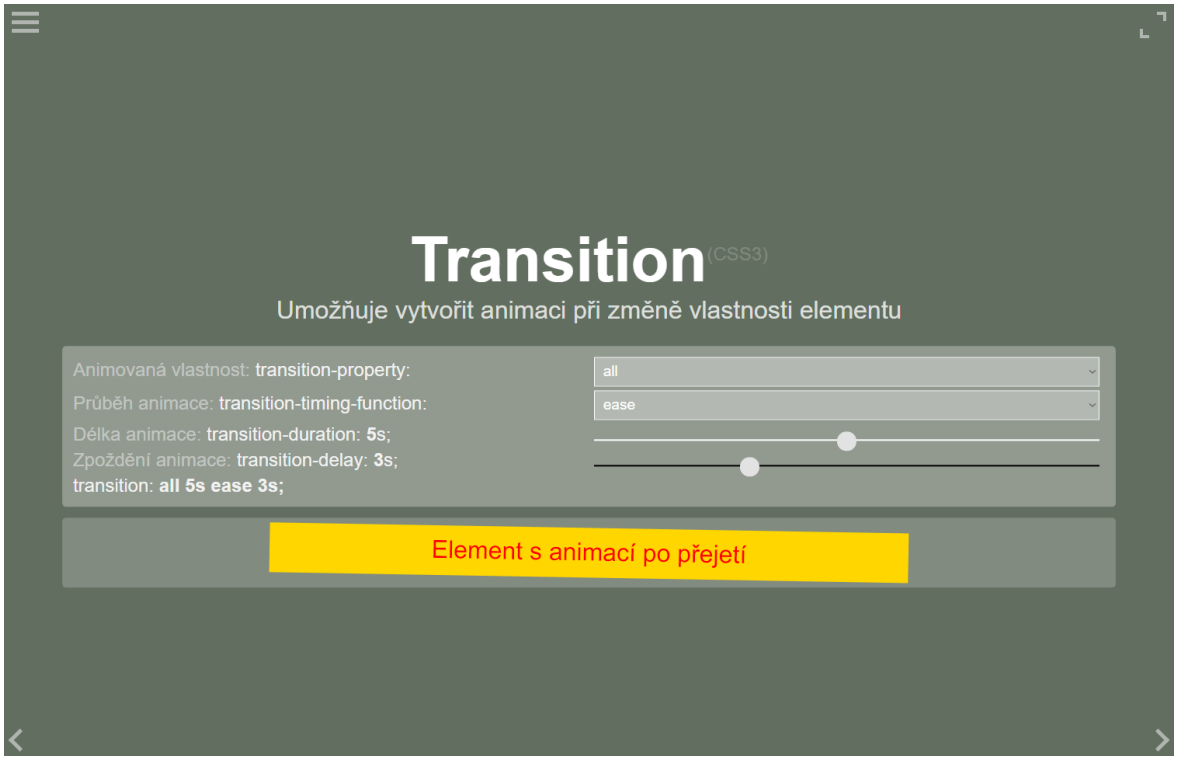
36) Quick start. Methodology / BEM [online]. [cit. 2018-03-28]. Dostupné z: https://en.bem.info/methodology/history/

37) Selectors Level 4. *CSS Working Group Wiki* [online]. 2018 [cit. 2018-04-08]. Dostupné z: https://drafts.csswg.org/selectors-4/#rw-pseudos

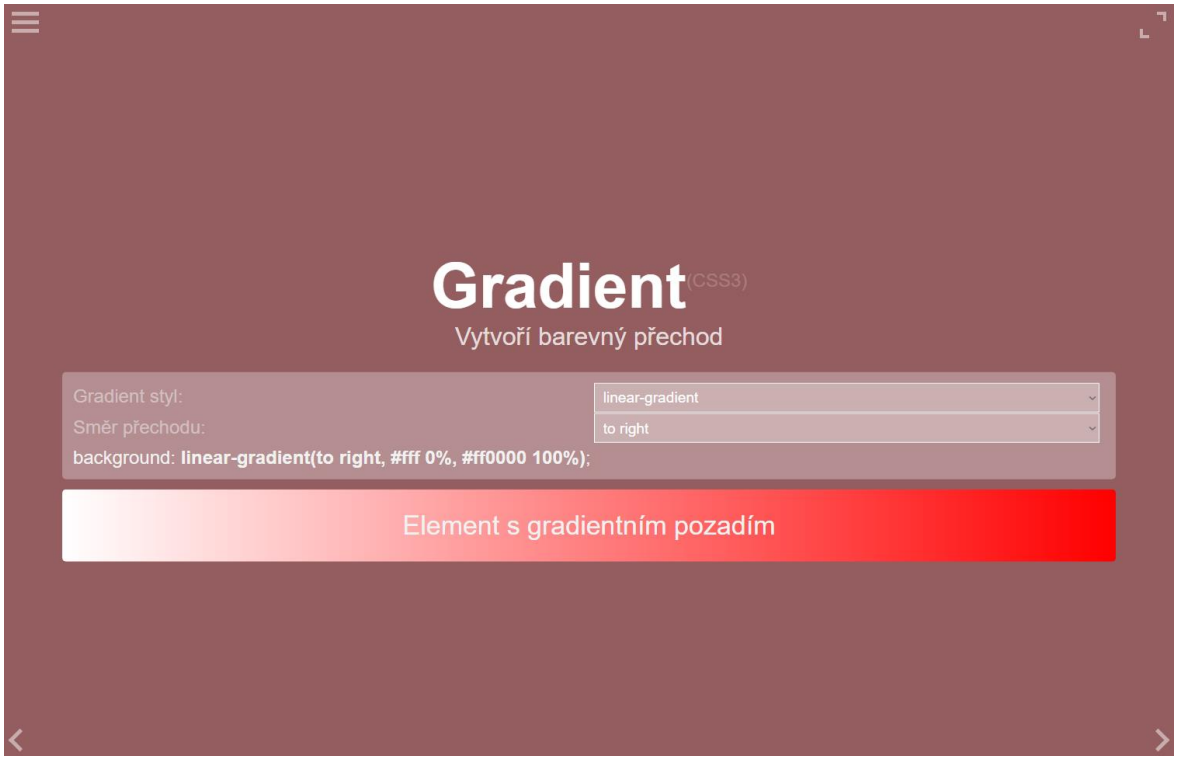
38) SHARKIE, Craig a Andrew FISHER. *Responzivní webdesign: okamžitě*. Brno: Computer Press, 2015. ISBN 978-80-251-4384-1.
39) SCHENGILI-ROBERTS, Keith. *Core CSS: cascading style sheets*. 2nd ed. Upper Saddle River, NJ: Prentice Hall PTR, c2004. ISBN 0-13-009278-9.
40) SNOOK, Jonathan. *Scalable and modular architecture for CSS*. 2nd ed: Snook.ca Web Development, 2012. ISBN 9780985632106.
41) SPŁAWSKI, Jacek. Everything about Sass - what it gives us and why to choose it. *Merixstudio* [online]. 26. 11. 2015 [cit. 2018-04-02]. Dostupné z: https://www.merixstudio.com/blog/everything-about-sass-what-it-gives-us-and-why-choose-it/
42) SUEHRING, Steve. *JavaScript: krok za krokem*. Brno: Computer Press, 2008. Krok za krokem (Computer Press). ISBN 978-80-251-2241-9.
43) TEAGUE, Jason Cranford. *DHTML a CSS pro World Wide Web: praktická vizuální příručka.* Praha: SoftPress, c2005. ISBN 80-86497-77-1.
44) Top 10 Front-End Frameworks of 2016. *KeyCDN* [online]. 5. 2. 2018 [cit. 2018-04-01]. Dostupné z: https://www.keycdn.com/blog/front-end-frameworks/
45) VÁCLAVEK, Petr. Designové trendy pro rok 2015. *Ilustrátor* [online]. 24. 6. 2015 [cit. 2018-04-17]. Dostupné z: http://ilustrator.cz/designove-trendy-rok-2015/
46) W3.CSS Tutorial. *W3schools* [online]. [cit. 2018-04-01]. Dostupné z: https://www.w3schools.com/w3css/default.asp
47) Webové trendy pro rok 2016 - Začínáme s HTML. *WebDev* [online]. 23. 06. 2016 [cit. 2018-03-21]. Dostupné z: https://cs.webdev.wiki/howto/webove-trendy-pro-rok-2016.html
48) What are Frameworks? 22 Best Responsive CSS Frameworks for Web Design. AWWWARDS [online]. [cit. 2018-03-29]. Dostupné z: https://www.awwwards.com/what-are-frameworks-22-best-responsive-css-frameworks-for-web-design.html
49) What's New in Bootstrap 3 Related Topics:. *Sitepoint* [online]. 5. 9. 2013 [cit. 2018-03-30]. Dostupné z: https://www.sitepoint.com/whats-new-bootstrap-3/

## Seznam příloh

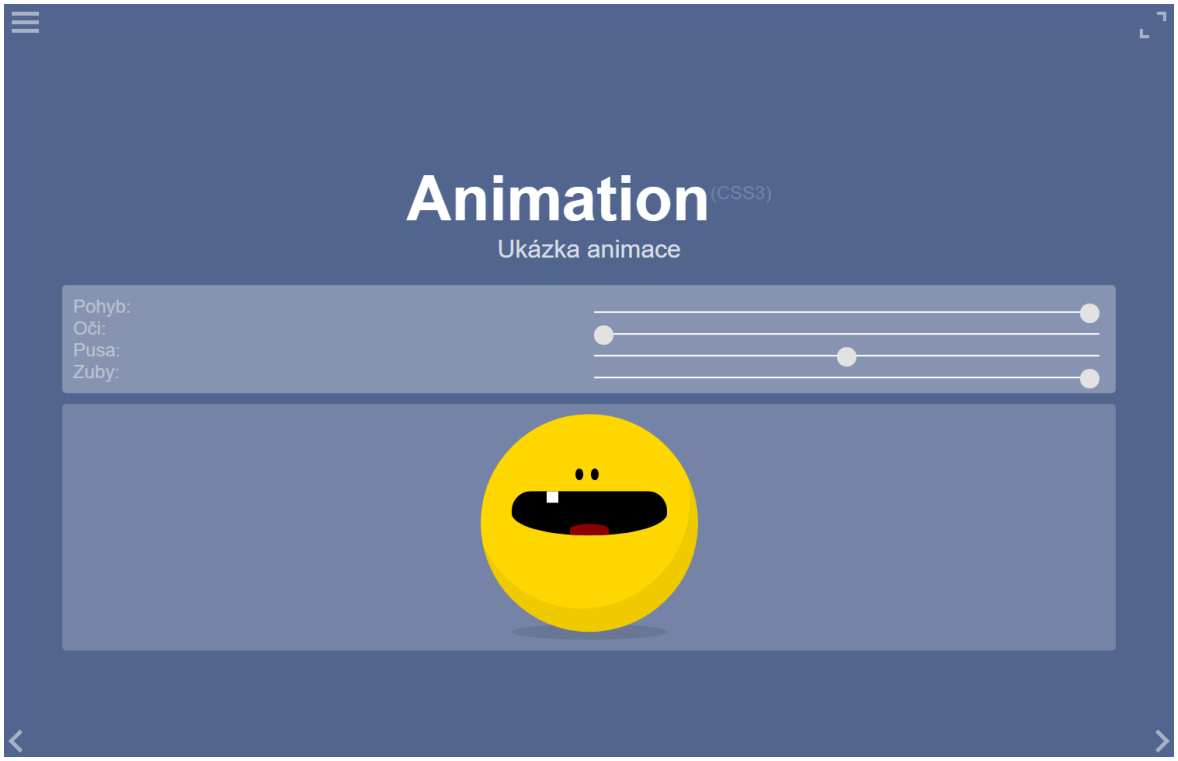
## Příloha 2 – Ukázka z webové prezentace



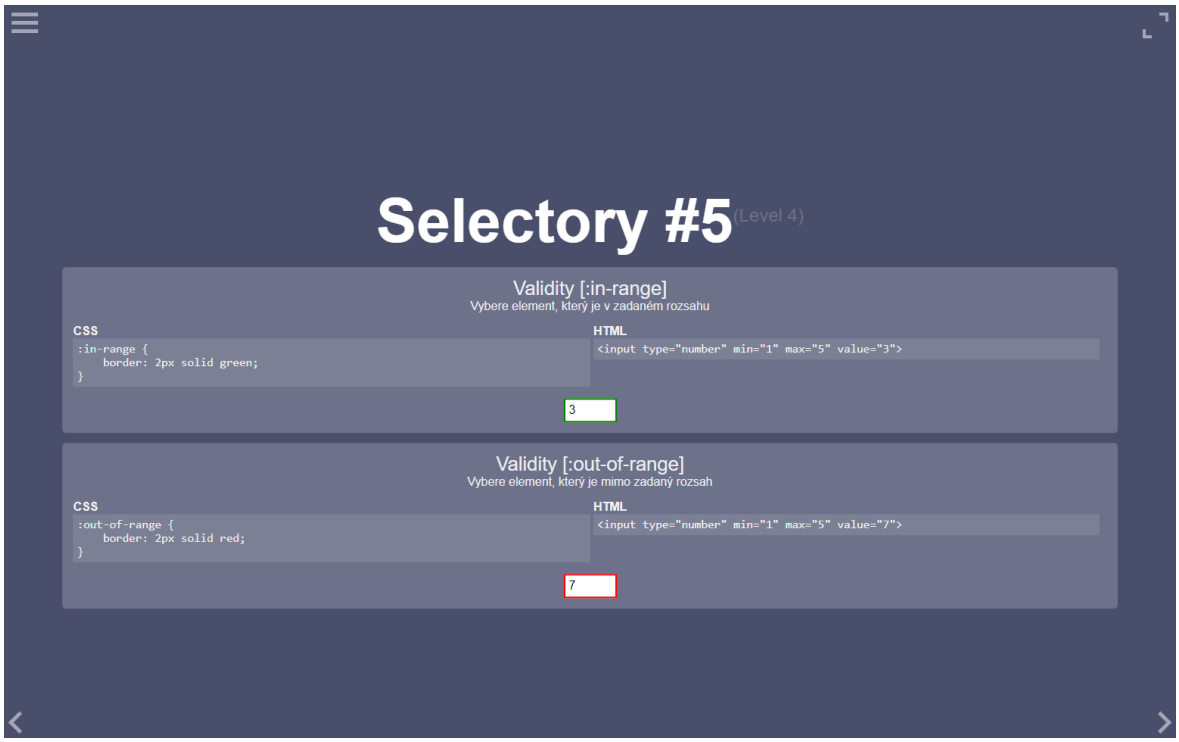

*Obrázek 9 Ukázka vlastnosti transition v prezentaci*




*Obrázek 10 Ukázka přechodů v prezentaci*




*Obrázek 11 Ukázka animace v prezentaci*




*Obrázek 12 Ukázka nových selektorů v prezentaci*

**Příloha 3 - Podpora CSS vlastností v prohlížečích**

***Podpora sloupců v prohlížečích***

| | IE | Chrome | Firefox | Opera | Safari |
|---|---|---|---|---|---|
| Podpora od verze | 10.0 | 50.0 | 52.0 | 37 | 9.0 |
| Podpora s prefixem od verze | | 4.0 | 2.0 | 11.1 | 3.1 |

Zdroj: CSS Multiple Columns. *W3 schools* [online]. [cit. 2018-04-05]. Dostupné z: https://www.w3schools.com/css/css3_multiple_columns.asp

***Podpora vlastnosti text-shadow v prohlížečích***

| | IE | Chrome | Firefox | Opera | Safari |
|---|---|---|---|---|---|
| Podpora od verze | 10.0 | 4.0 | 3.5 | 9.6 | 4.0 |

Zdroj: CSS text-shadow Property. *W3 schools* [online]. [cit. 2018-04-05]. Dostupné z: https://www.w3schools.com/css/css3_pr_text-shadow.asp

***Podpora vlastnosti box-shadow v prohlížečích***

| | IE | Chrome | Firefox | Opera | Safari |
|---|---|---|---|---|---|
| Podpora od verze | 9.0 | 10.0 | 4.0 | 10.5 | 5.1 |
| Podpora s prefixem od verze | | 4.0 | 3.5 | 11.1 | 3.1 |

Zdroj: CSS box-shadow Property. *W3 schools* [online]. [cit. 2018-04-05]. Dostupné z: https://www.w3schools.com/css/css3_pr_box-shadow.asp

***Podpora vlastnosti border-radius v prohlížečích***

| | IE | Chrome | Firefox | Opera | Safari |
|---|---|---|---|---|---|
| Podpora od verze | 9.0 | 5.0 | 4.0 | 10.5 | 5.0 |
| Podpora s prefixem od verze | | 4.0 | 3.0 | 11.1 | 3.1 |

Zdroj: CSS border-radius Property. *W3 schools* [online]. [cit. 2018-04-05]. Dostupné z: https://www.w3schools.com/css/css3_pr_border-radius.asp

***Podpora vlastnosti opacity v prohlížečích***

| | IE | Chrome | Firefox | Opera | Safari |
|---|---|---|---|---|---|
| Podpora od verze | 9.0 | 4.0 | 2.0 | 9.0 | 3.1 |
| Podpora s prefixem od verze | Pro starší verze je možné použít filter: alpha(opacity=X) | | | | |

Zdroj: CSS opacity Property. *W3 schools* [online]. [cit. 2018-04-05]. Dostupné z: https://www.w3schools.com/css/css3_pr_opacity.asp

***Podpora vlastnosti transform v prohlížečích***

| | IE | Chrome | Firefox | Opera | Safari |
|---|---|---|---|---|---|
| Podpora od verze (2D) | 10.0 | 36.0 | 2.0 | 9.0 | 9.0 |
| Podpora s prefixem od verze (2D) | 9.0 | 4.0 | 3.5 | 10.5 | 3.2 |
| Podpora od verze (3D) | 12 | 36.0 | 10.0 | 23.0 | 9.0 |
| Podpora s prefixem | | 12.0 | | 15.0 | 4.0 |

| | | | | | |
|---|---|---|---|---|---|
| od verze (3D) | | | | | |

Zdroj: CSS transform Property. *W3 schools* [online]. [cit. 2018-04-05]. Dostupné z: https://www.w3schools.com/css/css3_pr_transform.asp

***Podpora vlastnosti perspective v prohlížečích***

| | IE | Chrome | Firefox | Opera | Safari |
|---|---|---|---|---|---|
| Podpora od verze | 10.0 | 36.0 | 16.0 | 23.0 | 9.0 |
| Podpora s prefixem od verze | | 12.0 | 10.0 | 15.0 | 4.0.3 |

Zdroj: CSS perspective Property. *W3 schools* [online]. [cit. 2018-04-05]. Dostupné z: https://www.w3schools.com/css/css3_pr_perspective.asp

***Podpora CSS vlastnosti filter v prohlížečích***

| | IE | Chrome | Firefox | Opera | Safari |
|---|---|---|---|---|---|
| Podpora od verze | 13.0 | 53.0 | 35.0 | 40.0 | 9.1 |
| Podpora s prefixem od verze | | 18.0 | | 15.0 | 6.0 |

Zdroj: CSS filter Property. *W3 schools* [online]. [cit. 2018-04-05]. Dostupné z: https://www.w3schools.com/css/css3_pr_filter.asp

***Podpora přechodů v prohlížečích***

| | IE | Chrome | Firefox | Opera | Safari |
|---|---|---|---|---|---|
| Podpora od verze | 10.0 | 26.0 | 16.0 | 12.1 | 6.1 |
| Podpora s prefixem od verze | | 10.0 | 3.6 | | 5.1 |

Zdroj: Can I use linear-gradient. *Can I use* [online]. [cit. 2018-04-06]. Dostupné z: https://caniuse.com/#search=radial-gradient

***Podpora transition v prohlížečích***

| | IE | Chrome | Firefox | Opera | Safari |
|---|---|---|---|---|---|
| Podpora od verze | 10.0 | 26.0 | 16.0 | 12.1 | 6.1 |
| Podpora s prefixem od verze | | 4.0 | 4.0 | 10.5 | 3.1 |

Zdroj: CSS Transitions. *W3 schools* [online]. [cit. 2018-04-05]. Dostupné z: https://www.w3schools.com/css/css3_transitions.asp

***Podpora animací v prohlížečích***

| | IE | Chrome | Firefox | Opera | Safari |
|---|---|---|---|---|---|
| Podpora od verze | 10.0 | 43.0 | 16.0 | 30.0 | 9.0 |
| Podpora s prefixem od verze | | 4.0 | 5.0 | 12.0 | 4.0 |

Zdroj: CSS Animations. *W3 schools* [online]. [cit. 2018-04-05]. Dostupné z: https://www.w3schools.com/css/css3_animations.asp

***Podpora nových selektorů v prohlížečích***

| | IE 9 | IE 10 | IE 11 | MS Edge 41.16 | Firefox 59 | Chrome 65 |
|---|---|---|---|---|---|---|
| Case sensitivity | × | × | × | × | ✓ | ✓ |
| Placeholder | × | ✓ | ✓ | ✓ | ✓ | ✓ |
| Optionality | × | ✓ | ✓ | ✓ | ✓ | ✓ |
| Mutability | × | × | × | ✓ | × | ✓ |

| | | | | | | |
|---|---|---|---|---|---|---|
| Range | × | ✓ | ✓ | ✓ | ✓ | ✓ |
| Validity | × | × | × | ✓ | ✓ | ✓ |
| Indeterminate | × | ✓ | ✓ | ✓ | ✓ | ✓ |
| Default | × | × | × | × | ✓ | ✓ |