Vysoká škola ekonomická v Praze

Fakulta informatiky a statistiky

Katedra informačního a znalostního inženýrství

CSS/JS knihovna pro zjednodušení vývoje grafického rozhraní webových stránek

BAKALÁŘSKÁ PRÁCE

ve studijním programu Aplikovaná informatika

Autor: Martin Škára

Vedoucí: Ing. et Ing. Stanislav Vojíř, Ph.D.

Praha, duben 2018

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a že jsem uvedl všechny použité prameny a literaturu, ze které jsem čerpal.

V Praze dne 27. 4. 2018 . . . . . .

Martin Škára

Poděkování

Chtěl bych poděkovat vedoucímu práce Ing. Et Ing. Stanislavovi Vojířovi, Ph. D. za odbornou konzultační činnost při tvorbě práce. Dále bych chtěl poděkovat i kolegům z firmy Appio za cenné poznatky při tvorbě praktické části práce.

Abstrakt

Jednou z částí vývoje, kterou musí projít každá webová stránka, je tvorba grafického rozhraní. Jelikož ale na stránkách můžeme často najít společné prvky či styly, vytvářet tak vše stále znovu by bylo velmi časově náročné a tudíž kontraproduktivní. I proto vznikají znovupoužitelné knihovny, které vývoj grafického rozhraní zjednodušují a definují vzhled a chování často užívaných prvků stránky. Vývojář tak může přemýšlet nad tím, kterou z těchto knihoven zvolit, či zda případně vyvinout nástroj vlastní, který bude plně vyhovovat jeho požadavkům. Hlavním cílem práce je vývoj vlastní knihovny zjednodušující vývoj grafického rozhraní obsahující grid systém, pomocné třídy a znovupoužitelné komponenty. Samotná knihovna byla vyvíjena v CSS preprocesoru SASS a javascriptové knihovně jQuery. Práce popisuje jak samotný vývoj knihovny, tak i použité technologie, analýzu existujících řešení pro tvorbu grafického rozhraní webových stránek, vytvoření komplexních příkladů demonstrující možnosti knihovny, vytvoření dokumentace a publikování knihovny.

Klíčová slova

CSS knihovna, grafické rozhraní, webová stránka, CSS, SASS, jQuery

Abstract

The important part of website development is the development of a graphical interface. On most websites, however, we can find similar elements or styles and developing all parts again and again would be really time consuming and contra productive. That is the reason why re-usable libraries are created to simplify the development of the graphical interface and often they define the basic appearance of the websites elements. The developer can then think about which of these libraries he will use or if he will develop a custom tool that will fully meet his requirements. The bachelor thesis deals with the development of a custom library simplifying the development of the graphical interface. The library includes grid system, helper classes and re-usable components. The library itself was developed in CSS preprocessor SASS and jQuery JavaScript library. The thesis describes the development of the library itself, technologies that were used, analysis of existing solutions for development of a graphical interface, creation of complex examples demonstrating the possibilities of the library, creation of documentation and the publishing of the library. The development of the library itself is the main goal of the entire bachelor thesis.

Keywords

CSS library, graphical interface, website, CSS, SASS, jQuery

Obsah

Úvod

Jednou z částí vývoje, kterou musí projít každá webová stránka, je tvorba grafického rozhraní. Jelikož ale na stránkách můžeme často najít společné prvky či styly, vytvářet tak vše stále znovu by bylo velmi časově náročné a tudíž kontraproduktivní. I proto vznikají znovupoužitelné knihovny, které vývoj grafického rozhraní zjednodušují a často právě stránce definují i základní vzhled. Pro vývojáře může tak vyvstat otázka, kterou z těchto knihoven zvolit či případně vyvinout nástroj vlastní, který bude plně vyhovovat jeho požadavkům.

Autor práce je zaměstnancem firmy Appio Digital s.r.o. (dále Appio), kde působí jako webový kodér. Věnuje se tak zejména vývoji uživatelského a grafického rozhraní. Na starost má zejména projekty menšího rozsahu, které běží na PHP frameworku Symfony (https://symfony.com/). Nástroje pro tvorbu grafického rozhraní si pak určují sami kodéři. Momentální situace je taková, že každý z projektů je tvořen pomocí jiné knihovny či vlastních nástrojů jednotlivých vývojářů a některé z komponent webových stránek (např. menu) nejsou vytvářeny jako znovupoužitelné, což brzdí vývoj zejména z časového hlediska. Bylo by tak vhodné používat pouze jeden stálý nástroj pro tvorbu takových projektů. Jelikož autorovi práce nevyhovují již hotové knihovny, rozhodl se vyvinout nástroj vlastní, který bude čerpat z již vytvořených projektů a bude tak korespondovat s požadavky autora i vývoje v jeho zaměstnání. Jeho cílem nebude, na rozdíl od většiny hotových knihoven, náhrada grafického rozhraní, ale měl by pomoci s vývojem rozhraní dle různorodého grafického návrhu.

Hlavním cílem práce je vyvinout vlastní knihovnu, která usnadní tvorbu grafického rozhraní dle různorodého grafického návrhu. Hlavní cíl je dále rozdělen na následující dílčí cíle (DC):

- Prvním dílčím cílem (DC1) je vytvoření grid systému (struktury sloužící k pozicování jednotlivých prvků a lze díky ní tak snadno řadit prvky do řádků či sloupců). [1]
- Druhým dílčím cílem (DC2) je vytvoření znovupoužitelných komponent, které budou vycházet z běžných prvků objevujících se často na webových stránkách.
- Třetím dílčím cílem (DC3) je vytvoření pomocných tříd, které umožní vývojářům urychlit další vývoj částí stránek, které ze samotné knihovny vycházet nebudou.
- Čtvrtým dílčím cílem (DC4) je vytvoření dokumentace pro všechny vytvořené části knihovny.

- Posledním, pátým dílčím cílem (DC5), je vytvoření modelových stránek pro demonstraci možností implementované knihovny, které budou používat grid systém, hotové komponenty i pomocné třídy vytvořené knihovny.

Z hlediska struktury práce jsou v první kapitole popsány technologie, které autor při tvorbě knihovny využil. V druhé kapitole jsou rozebrány již hotová řešení (knihovny), jež se autor snaží analyzovat zejména z hlediska potřeb budoucích projektů a některými těmito řešeními se nechává dále inspirovat v samotné tvorbě knihovny. Kapitoly 3 a 4 se věnují již návrhu a vývoji samotného nástroje. V kapitole 5 se pak autor věnuje vytvoření modelových stránek a příkladů, s jejichž pomocí chce demonstrovat možnosti knihovny. Kapitola 6 se zabývá vytvořením dokumentace a publikováním knihovny. Práce tak může sloužit jako příklad pro další vývojáře, kteří by se chtěli pustit do vývoje obdobné knihovny. Autor předpokládá, že čtenář má alespoň základní povědomí o vytváření webových stránek.

1. Použité technologie

V této kapitole jsou popsány technologie, které byly použity při vytváření samotné knihovny (z nichž většinu používají i již hotová alternativní řešení, popsána v kapitole 2). Patří mezi ně *CSS* a jeho preprocesor *SASS*, *JavaScript* a jeho knihovna *jQuery* a nástroje pro zkompilování knihovny – *Yarn* a *Gulp*.

1.1. CSS

CSS (Cascading Style Sheets) je jazyk, který ovlivňuje vlastnosti zobrazení prvků v souborech vytvořených pomocí jazyka *HTML*, který slouží pro zápis struktury webových stránek a dalších souborů. Jeho hlavním smyslem je oddělení vzhledových vlastností od obsahu jednotlivých prvků. Jeho současná verze je CSS3, starší verze nabízely daleko méně možností a vlastností. Již od první verze CSS byla zavedena syntaxe vybírání prvků pomocí selektorů a nastavování jejich hodnot. Nastavovat šly vlastnosti písma a textu, barvy textu i pozadí, rozměry elementů, jejich pozice apod. [2]

Zápis CSS může vypadat například takto:

```
.text { /*selektor třídy*/
  color: black;
  font-size: 12px;
  font-weight: bold;
}
```

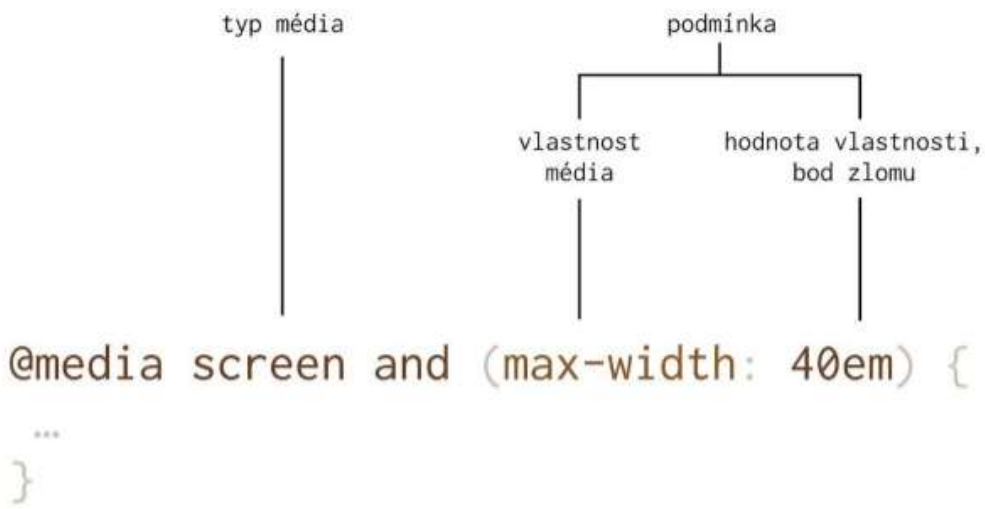
Kód 1 - Ukázka zápisu třídy v jazyce CSS [zdroj autor]

Důležitou funkcí CSS je v dnešní době tvorba rozložení (layoutu) stránky. V minulosti se pro tvorbu rozložení stránky používaly HTML rámy či tabulky. V dnešní době se pro tvorbu rozložení využívají vlastnosti CSS, HTML slouží pouze k znázornění struktury obsahu. Jak píše Rachel Andrew v knize CSS3 Layout Modules ([5]), před moderními přístupy, které jsou zmíněny dále, se využívalo zejména proměnlivých rozměrů prvků a vlastností `float` a `display: inline-block`, absolutního pozicování či `display: table`. `Float` je původně vlastnost určující obtékání prvku, často se využívá pro zarovnání prvků vedle sebe, čehož se využívá v rámci tvorby grid systémů, stejně jako vlastnosti `display: inline-block`. Žádná z těchto vlastností však nebyla původně navržena pro tvorbu rozložení stránky, tudíž se dnes čím dál více přistupuje k novějším CSS modulům. Těmi jsou konkrétně *flexbox* a *CSS Grid* (modul jazyka CSS, nikoliv grid systém). Pro

tvorbu responzivního layoutu, kdy se stránka navrhuje tak, aby se přizpůsobila zařízení uživatele, jsou také důležitá pravidla *Media Queries*.[3] [4] [5]

1.1.1. Media Queries

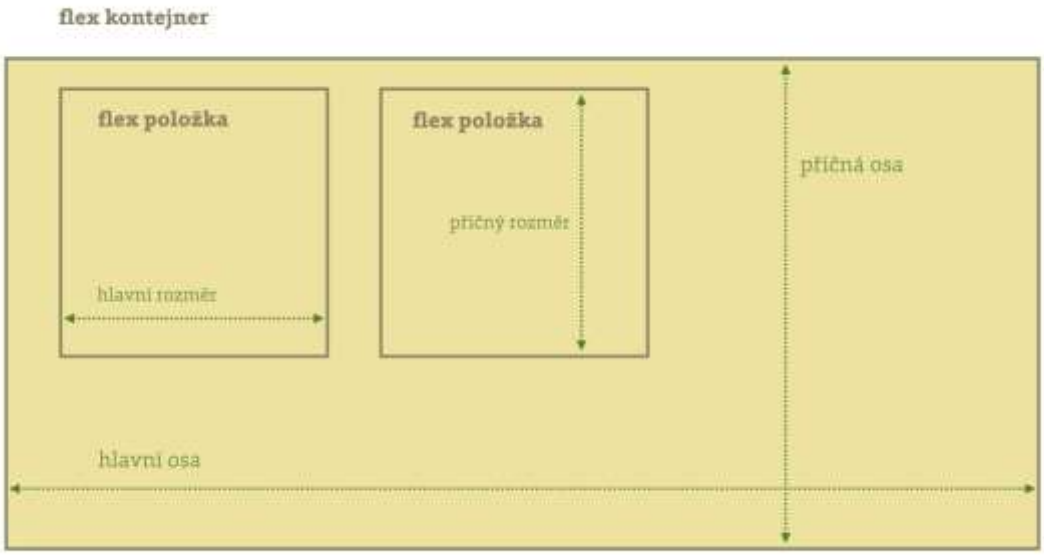
Specifikace Media Queries udává v rámci jazyka podmínky, které umožní aplikovat určité CSS vlastnosti v určitých situacích. Dnes slouží zejména pro definování CSS dle velikosti okna prohlížeče a orientace zařízení. Jsou jedním z nejdůležitějších nástrojů pro tvorbu responzivního layoutu. Celkově lze jimi ale definovat i styly pro monitory s určitým rozlišením a poměrem, pro tisk a pro prohlížeče a zařízení se specifickými vlastnostmi. Typickou zjednodušenou strukturu Media Query lze vidět na obrázku níže. [6]



Obrázek 1 - Zjednodušená struktura Media Query zápisu. [6]

1.1.2. Flexbox

Flexbox je modul CSS, jehož vlastnosti určují rodičovskému prvku (Flex container, flex kontejner) jak velké místo v něm budou zaujímat a jak se budou zarovnávat prvky v něm obsažené (Flex items, flex položky). Schéma lze vidět dále na obrázku 2. Ve výchozím nastavení manipuluje s prvky tak, že se snaží přizpůsobit jejich vzájemné rozměry jejich potřebnému obsahu. Prvky jsou tak flexibilní. [7]



Obrázek 2- Schéma použití flexboxu [7]

Flex kontejneru lze nastavit následující vlastnosti: [7]

- `Flex-direction` – směr v jakém se uvnitř kontejneru budou řadit položky – definuje hlavní osu. Defaultně zleva do prava.
- `Flex-wrap` – zalamování řádků / sloupců položek, pokud jejich rozměry přesáhnou rozměr kontejneru. Defaultně se nezalamují.
- `Justify-content` – zarovnávání prvků ve směru hlavní osy. Defaultně se zarovnávají na začátek.
- `Align-items` – zarovnávání prvků v rámci příčné osy (tj. osy kolmé k ose hlavní). Defaultně se položky v kontejneru chovají tak, aby jejich příčný rozměr byl stejný jako příčný rozměr kontejneru.
- `Align-content` – zarovnávání řádků (či sloupců v případě, že je hlavní osou vertikální osa) v rámci kontejneru. Defaultně si řádky rozměry rovnoměrně vyplní tak, aby zaplnily celý příčný rozměr kontejneru.

Jednotlivým flex položkám pak lze nastavit následující vlastnosti: [7]

- `Order` – určuje pořadí položky v rámci flex kontejneru
- `Flex-grow` – určuje schopnost položky zvětšit svůj rozměr oproti ostatním položkám, pokud je to nutné

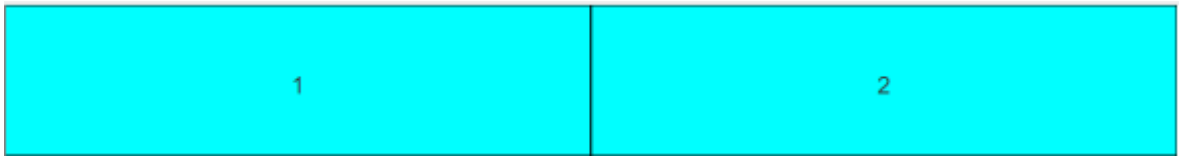
- `Flex-shrink` – určuje schopnost položky zmenšit svůj rozměr oproti ostatním položkám, pokud je to nutné
- `Flex-basis` – výchozí rozměr položky, předtím, než si položky rozdělí místo v kontejneru dle vlastností `flex-grow` a `flex-shrink`
- `Align-self` – přetěžuje vlastnost `align-items` v kontejneru, ale pouze právě u jedné položky.

I přes jednoduché použití vyčtených vlastností se stále k flexboxu mezi vývojáři přistupuje s jistou obezřetností. Důvodem jsou problémy a podpora flexboxu v různých prohlížečích. Autor práce je však toho názoru, že obavy již dnes nejsou na místě. Pokud se jako vývojář budeme zajímat pouze o prohlížeče, které používá alespoň 0,5 % z celkového počtu všech uživatelů webových stránek, má dnes (březen 2018) flexbox problém pouze v rámci prohlížeče Internet Explorer 11, kde některé jeho vlastnosti v určitých situacích nefungují správně. Například v případě použití vlastnosti `flex` je třetí hodnotou vyžadována jednotka či v případě použití vlastnosti `flex-basis` je potřeba prvku přiřadit i vlastnosti `max-width` se stejnou hodnotou, aby se prvky v řádku správně zalamovaly v případě, že přesahují flex kontejner. [8]

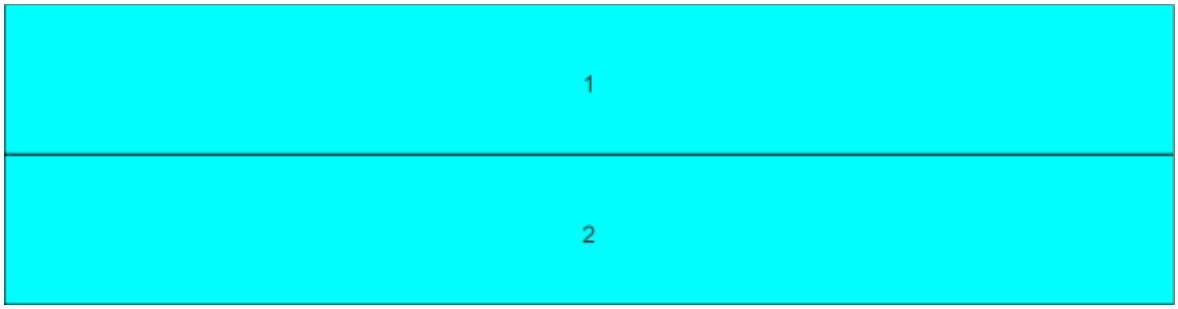
1.1.3. Grid systémy

S rozmachem responzivního webdesignu je jedním z důležitých součástí webových stránek grid systém, který však není technologií, ve které by byly knihovny vytvářeny, ale je jednou ze součástí knihoven zjednodušujících vývoj rozložení stránky. I tak je zcela na místě vysvětlit běžné chování takového systému. Představme si prvky (sloupce), které chceme zarovnávat do řádku. První z nich zabírá dvě třetiny tohoto řádku, druhý zbylou třetinu (obrázek 3) – nicméně pouze při velkém rozlišení obrazovky. Pokud si však uživatel zobrazí webovou stránku na menším displeji (např. tabletu), chceme, aby každý z prvků zabíral půl řádku (obrázek 4), pokud ještě na menším (např. mobilním telefonu), tak budou prvky zabírat celý řádek a budou zarovnány pod sebou (obrázek 5). Zápisy tohoto příkladu v jazyku HTML, jsou pomocí jednotlivých knihoven ukázány v kapitole 2. Jednotlivé body rozlišení obrazovky, kdy se chování sloupců (a případně i další chování stránky) mění, nazýváme *breakpointy*. Definují se zejména pro určení oblastí rozlišení, ve kterých bude web vypadat přibližně stejně. Typické rozdělení takových oblastí je oblast pro mobilní telefony, tablet a desktopová rozlišení.

Obrázek 3 - Zobrazení uvažovaných prvků na velkém rozlišení obrazovky [zdroj autor]



Obrázek 4 - Zobrazení uvažovaných prvků při menším rozlišení obrazovky [zdroj autor]



Obrázek 5 - Zobrazení uvažovaných prvků na nejmenším rozlišení obrazovky (např. mobilním telefonu) [zdroj autor]

Responzivní grid tak umožňuje vývojáři manipulovat s velikostí a uspořádáním prvků v závislosti na uživatelově rozlišení obrazovky (a velikosti okna prohlížeče).

1.2. CSS Preprocesory

Preprocesory jsou jazyky postavené nad CSS, které je následně nutné zkompilovat právě do CSS souborů. Využívány jsou proto, že na rozdíl od jazyka CSS nabízejí více funkcí a zjednodušují díky tomu práci. Mezi nejznámější preprocesory patří SASS (https://sass-lang.com/), mimo něj pak LESS (http://lesscss.org/) a Stylus (http://stylus-lang.com/). I přesto, že se jednotlivé preprocesory mohou lišit syntaxí a odlišností některých funkcí, lze říci, že téměř každý nabízí výsledné funkce:

- Ukládání hodnot do proměnných
- Zanořování selektorů
- Skládání více souborů dohromady
- Mixiny (skupiny vlastností s parametry, které lze následně používat v rámci celého kódu)
- Dědění tříd a vlastností

- Další funkce (práce s barvami, operátory atd.)

Lze namítnout, že pro některé z těchto funkcí existuje i specifikace v CSS, a tak by nebylo nutné používat preprocesor, například proměnné (https://www.w3.org/TR/css-variables-1/), skládání souborů, práce s operátory v rámci počítacích funkcí. Avšak s funkcemi preprocesorů lze daleko lépe pracovat a v některých případech (jako právě u již zmiňovaných proměnných v CSS, které například nefungují v prohlížeči Internet Explorer aj.) fungují po kompilaci na všech prohlížečích. Autor sám si vybral pro vývoj knihovny preprocesor SASS, zejména z toho důvodu, že se využívá při vývoji projektů ve firmě Appio a těší se všeobecně velké oblibě. [7] [9]

1.2.1. Preprocesor SASS

Preprocesor SASS (Syntactically Awesome Style Sheets) vznikl již roku 2006 a již od začátku je vyvíjen jako *open-source* pod licencí MIT (https://opensource.org/licenses/MIT), tj. kdokoliv jej pod touto licenci může používat, upravovat či vylepšovat, avšak není zaručena záruka funkčnosti. Jeho součástí jsou všechny funkce jmenované na začátku kapitoly 1.2. SASS, které se samozřejmě neustále vyvíjí, největší změnou a novinkou, kterou lze zmínit, je vyvinutí jeho syntaxe. Na počátku mělo SASS imperativní syntaxi. Ta je samotná dnes nazývaná jako SASS (stejně jako celý preprocesor). Stále ji lze používat, ale většina moderních knihoven v ní již není dále psána. [9] [10]

Syntaxe SASS vypadá následně:

```
.trida
  color: #fff
  width: 200px
```

Kód 2 - Ukázka syntaxe SASS preprocesoru SASS [zdroj autor]

S verzí 3 pak nabídl SASS (preprocesor) i syntaxi zvanou *SCSS*. Pokud jako ve výše uvedené ukázce zápisu nepoužijeme žádné funkce preprocesorů, vypadá pak jako CSS zápis. [9]

```
.trida {
  color: #fff;
  width: 200px;
}
```

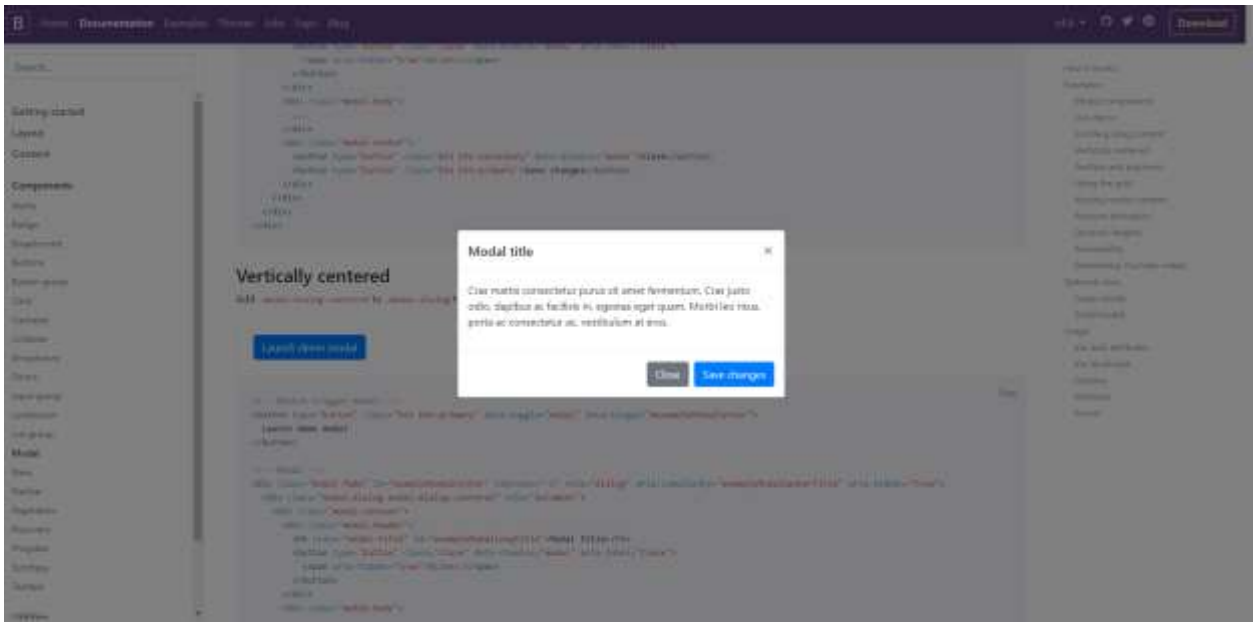
Kód 3 - Ukázka syntaxe SCSS preprocesoru SASS [zdroj autor]

V zásadě je ale jedno, jaká syntaxe je použita, jelikož se tím nijak nemění možnost funkcí. Pro tvorbu knihovny autor použil syntaxi SCSS. [9]

1.3. JavaScript

JavaScript je skriptovací jazyk, jehož kód se spouští v prohlížeči uživatele a umožňuje s webovou stránkou manipulovat a reagovat na jeho akce. V rámci vytvořené knihovny je použit zejména kvůli manipulaci se třídami elementů a pro vytvoření dynamických komponent knihovny, např. *carouselu* (Obrázek 6) či modálního okna (Obrázek 7). Pro zjednodušení je použita knihovna jQuery. [2]

Obrázek 6 - Ukázka komponenty Carousel vytvořené pomocí knihovny Bootstrap. Carousel je všeobecně uznávaný název pro tento druh komponenty. [11]



Obrázek 7 - Ukázka komponenty modálního okna vytvořené pomocí knihovny Boostrap. Modal (či Modální okno) je všeobecně uznávaný název pro tento typ komponenty.[12]

1.3.1. jQuery

jQuery je javascriptová knihovna, která usnadňuje práci s HTML dokumentem. S její pomocí lze mnoha věcí, které by normálně psal vývojář v čistém JavaScriptu, dosáhnout daleko jednodušeji a přehledněji. Zde je seznam některých výhod této knihovny:

- jQuery je vyvíjeno jako open-source pod licencí MIT, tudíž je zdarma a volně šířitelné
- Je využíváno tvůrci webových stránek na celém světě a je velmi rozšířené. Je tak velká možnost, že v případě jeho načtení ze stejného zdroje[1] jej uživatel má již načtené v prohlížeči a nemusí tak knihovnu po navštívení další stránky znovu stahovat.
- Zaručuje širokou kompatibilitu napříč prohlížeči, vývojář se tak ve většině případů nemusí bát, že by v moderních prohlížečích jeho aplikace nefungovala
- Je k němu dostupná kompletní dokumentace

[1] Například z CDN (Content Delivery Network), V praxi to funguje tak, že požadovaná data nejsou umístěna jen na jednom serveru, ale jsou distribuována na více serverech po světě. Návštěvník, jehož prohlížeč začne požadovat určitý obsah, tak bude připojen na server, který má blíž. Od získávání dat z blíže umístěného serveru se očekává lepší odezva a vyšší rychlost. Z CDN jsou tak stahovány zejména knihovny, které jsou více rozšířené. [http://jecas.cz/cdn]

- Mnoho dalších knihoven jQuery využívá a je tak jednoduché najít knihovnu či plugin řešící vývojářův problém[2]
- Šetří zápis kódu – vytvořený kód je ve většině případů přehlednější, jednodušší pro pochopení

Knihovna využívá aktuální (duben 2018) verzi jQuery 3.3.1 slim. Slim verze jQuery, na rozdíl od plné verze, neobsahuje některé funkce a moduly a výsledný soubor knihovny je tak menší. [13] [14].

1.4. Nástroje pro zkompilování knihovny

Jelikož knihovna má být k dispozici k použití i v rámci jednoho CSS a jednoho JS souboru (pokud vývojář použije některé z komponent využívající JavaScript/jQuery), je nutné celou knihovnu zkompilovat, zejména z toho důvodu, že knihovna je v nezkompilovaném stavu tvořena více soubory. Mimo utvoření a sestavení těchto souborů proběhne během kompilace ještě několik operací:

- Kompilace SASS (resp. SCSS) souborů do validního CSS
- K některým problematičtějším vlastnostem CSS se přidají potřebné prefixy pomocí pluginu *Autoprefixer* (https://github.com/postcss/autoprefixer). Některé vlastnosti CSS, například `filter` jsou v některých prohlížečích totiž podporovány pouze s prefixem. Vlastnosti jsou pak definovány vícekrát – např. u vlastnosti `filter` jednak normálně a jednak jako vlastnost `-webkit-filter`. Toto zlepší kompatibilitu s prohlížeči, kde by vlastnost bez prefixu podporována nebyla.
- Výsledné CSS a JS soubory se minifikují (tj. zmenšují, odstraní se přebytečné znaky souborů, komentáře a zkracují zápisy některých funkcí a vlastností) aby se zmenšila jejich výsledná velikost. Minifikovaná verze knihovny nebude k dispozici, avšak vývojář může knihovnu sám zkompilovat tak, aby se minifikace neprovedla.

Kompilace knihovny by mohla být provedena mnoha způsoby, autor v rámci knihovny uvede příklad kompilace v nástroji Gulp, který všechny výše uvedené operace provede automaticky pomocí jednoho příkazu a potřebné nástroje a pluginy pak budou nainstalovány pomocí balíčkovacího

[2] Například třeba plugin pro Carousel

systému Yarn, který zase jedním příkazem dokáže tyto nástroje nainstalovat a získat jejich aktuální verze.

1.4.1. Yarn

Yarn je balíčkovací systém Node.js (https://nodejs.org), který čerpá balíčky a moduly ze stejného rozhraní, jako starší systém NPM (https://www.npmjs.com). Balíčky se instalují pomocí příkazů v příkazovém řádku a jejich struktura je také zapsána v souboru `package.json`. Yarn autor použil namísto NPM zejména proto, že instalace balíčků pomocí něj je rychlejší, zejména díky tomu, že Yarn instaluje balíčky paralelně – dle článku od Johna Nikhila je Yarn průměrně 4,7x rychlejší než NPM a jeho popularita stále roste. Na serveru GitHub je totiž již 2x oblíbenějších než NPM. [15] [16]

1.4.2. Gulp

Gulp je nástroj napsaný v JavaScriptu sloužící pro automatizaci úloh uvedených na začátku kapitoly 1.4. Je schopný všechny tyto úlohy spustit jedním příkazem zadaným do příkazové řádky. Je dostupný i jako balíček, takže jej lze nainstalovat pomocí systému Yarn. Úlohy, které chce vývojář automatizovat, se zapisují do souboru `gulpfile.js`, samotný Gulp se následně spouští z příkazové řádky. [17]

2. Analýza existujících knihoven pro řešení grafického rozhraní webových stránek

Před vlastním vytvářením knihovny pro řešení grafického rozhraní webových stránek je namístě nejdříve analyzovat hotová řešení, z nichž se lze poučit či inspirovat.

Mimo grid systém obsahují tyto knihovny většinou sadu hotových komponent pro webové stránky a pomocné třídy. Autor se tak v této kapitole zajímá zejména o tyto tři součásti takových knihoven a také o technologie, ve kterých byly tyto knihovny vytvořeny.

Pro analýzu si vybral následující knihovny, zejména z toho důvodu, že se s nimi buď již setkal či z důvodu jejich oblíbenosti mezi vývojáři:

- *Bootstrap* (https://getbootstrap.com/)
- *Foundation* (https://foundation.zurb.com/)
- *Pure CSS* (https://purecss.io/)
- *Bulma* (https://bulma.io/)

2.1. Bootstrap

Bootstrap byl původně vytvořen v roce 2010 vývojáři a designery ve společnosti Twitter (https://twitter.com/), již od začátku byl vyvíjen jako open-source software pod licencí MIT. Stejně jako naprostá většina podobných knihoven je jeho aktuální verze (4.0) vyvíjena v jazyku CSS, respektive jeho preprocesoru SASS. Existují však i verze v jiných preprocesorech. Pro některé své části využívá též JavaScript. Ten je v rámci Boostrapu využíván pomocí knihoven jQuery a Popper.js (https://popper.js.org/) pro správné fungování některých komponent. [12]

Bootstrap resetuje některé výchozí styly HTML prvků (převážně těch textových) a definuje jim vzhled bez toho aniž by vývojář de facto chtěl (bez přiřazení příslušných CSS tříd). Toto autor práce nepovažuje za správné, jelikož vzhled těchto prvků se často liší projekt od projektu, a tak je například zbytečné definovat barvu textu odstavce – upravením SCSS souborů lze toto sice změnit, nicméně vývojáře to stojí čas. Defaultně knihovna používá nativní fonty operačních systémů (dále systémové fonty) dle článku *Using UI System Fonts In Web Design: A Quick Practical Guide* od Marcina Wicharyho ([18]). Tím je tak docíleno toho, že defaultně není potřeba využívat žádný externí font

a zároveň je možno zachovat hezký výchozí vzhled na velké skupině zařízení. Zápis těchto nativních fontů v rámci CSS vlastnosti `font-family`, která definuje používaný font, může pak vypadat následně: [12] [18]

```
font-family: -apple-system, BlinkMacSystemFont,
             "Segoe UI", "Roboto", "Oxygen",
             "Ubuntu", "Cantarell", "Fira Sans",
             "Droid Sans", "Helvetica Neue", sans-serif;
```

Kód 4 - Zápis nativních fontů v rámci CSS vlastnosti font-family [18]

Font `-apple-system` označuje fonty San Francisco, Neue Helvetica a Lucida Grande ve webovém prohlížeči Safari na operačních systémech Mac OS X a iOS. O jaký font se jedná, rozhoduje verze operačního systému. Obdobně tyto fonty označuje font `BlinkMacSystemFont`, akorát na prohlížeči Google Chrome. Pokud se jedná o jiný operační systém, je použito Segoe UI pro Windows a Windows Phone, Roboto pro novější systémy Android a Chrome OS, Oxygen pro linuxové distribuce s grafickým prostředím KDE, Ubuntu pro linuxovou distribuci Ubuntu, Cantarell pro linuxové distribuce s grafickým prostředí GNOME, Fira Sans pro operační systém Firefox OS, Droid Sans pro starší verze systému Android, Helvetica Neue pro některé starší systémy iOS a pro zbytek systémů je pak použit font `sans-serif`. [18]

Pro velikost fontů (i jiných rozměrů) pak Bootstrap používá jednotky `rem`, které se odvozují od velikosti písma prvku `<html>`. Bootstrap nechává tuto velikost na 100 % písma v uživatelově prohlížeči (většinou 16 pixelů, pixel přestavuje jeden svítící bod na monitoru či v obrázku), od ní pak odvozuje zbylé velikosti. [12]

Pro velikosti grid systému jsou však použity pixely. Grid systém tvoří třída `container`, která definuje rozměry obsahu stránky a zarovnává jej na střed. V rámci ní lze použít třídu `row`, což je rodičovský prvek, v němž jsou následně definovány jednotlivé sloupce. Zápis příkladu z obrázků 3-5 pomocí Bootstrapu je ukázán v následujícím kódu. [12]

```
<div class="container">
  <div class="row>
    <div class="col-12 col-sm-6 col-lg-8">
      1
    </div>
    <div class="col-12 col-sm-6 col-lg-4">
      2
    </div>
  </div>
</div>
```

Kód 5 - Zápis uvažovaného příkladu z obrázků 3-5 pomocí knihovny Bootstrap [zdroj autor]

Sloupce tak mají povinný prefix `col-` za nímž následuje dvoumístná definice breakpointu. Boostrap má defaultně, ale s možností změny, definované takové body 4. Pokud se jedná o chování pod prvním breakpointem, nepíše se nic. Grid systém je *mobile-first*, což znamená, že defaultně se vzhled prvků definuje pro mobilní zařízení, a právě pomocí zmíněných breakpointů a CSS vlastností Media Queries k tomu sloužících se postupně definuje chování pro větší zařízení a displeje. Poslední číslo ve třídě pak znamená, kolik sloupců prvek zabírá. Pokud se číslo rovná maximálnímu počtu sloupců grid systému (v Bootstrapu defaultně nastaveno na 12, ale dá se změnit), roztáhne se prvek do šířky celého řádku. Mimo tyto třídy je ale možno použít i prostou třídu `col`, prvky si pak místo v řádku rozdělí zcela rovnoměrně. Mezery mezi sloupci jsou defaultně nastavené na 30 pixelů, pokud to vývojář nezmění. [12]

Grid však nabízí více možností – je totiž vytvořen pomocí CSS vlastností flexbox. Je tak možné jednoduše měnit zarovnávání prvků do řádků či sloupců, určovat, jak velký bude určitý Flex item oproti ostatním, jak se budou moci měnit flexibilně jeho rozměry, v jakém pořadí prvky řadit apod. [5]

Boostrap tak díky flexboxu nabízí rodičovskému prvku `row` třídy, které následně obsažené sloupce mohou vertikálně i horizontálně zarovnávat i různorodě v rámci zmíněných breakpointů. [12]

Pro jednotlivé sloupce pak nabízí třídy pro změnu pořadí prvku, či pro odsazení prvku, chování jde opět měnit dle rozlišení obrazovky. Tyto třídy považuje autor práce za zbytečné. Odsazení sloupce zleva či zprava není dle něj tak časté. Určité případy by se daly řešit pomocí tříd pro zarovnání v rodičovském prvku nebo vytvořením prázdného sloupce, který uživatel neuvidí. Stejně tak je to se změnou pořadí prvku, nejde o častý případ a často si lze vystačit pouze s tím, že prvek dáme na první či poslední místo. Bootstrap však nedefinuje třídy jen pro první a poslední místo v pořadí, ale pro tolik míst z kolika sloupců se skládá grid systém. I to považuje autor práce za nepřesné, jelikož i když bude grid systém definován jako dvanáctisloupcový, řádek může sloupců obsahovat více (řádek se pak zalomí). [12]

Z hlediska hotových komponent Bootstrap obsahuje většinu ze známých komponent, se kterými se lze na webových stránkách setkat. Za zmínku stojí menu a navigace, různá tlačítka, drobečková navigace (seznam odkazů pro nadřazené sekce), *carousel*, formulářové prvky a modální okna. Nevýhodou Bootstrapu je bohužel to, že i po úpravě těchto komponent může být vizuálně poznat, že vyšly právě z této knihovny – a tak je dnes v důsledku toho mnoho bootstrapových webových stránek velice podobných. [12]

Obrázek 8 - Typická ukázka webové stránky vytvořené pomocí bootstrapu [12]

U komponenty responzivního menu autor práce nalezl chybu. Pokud ji použijeme s fixní pozicí (tj. prvek zůstává na stále stejném místě v okně prohlížeče) a zobrazíme si ji na úzkém displeji (např. mobilním telefonu v zobrazení na šířku), tak v případě, že menu obsahuje více položek, uživatel některé neuvidí a nijak se k nim nedostane, což může být poněkud frustrující.

Bootstrap nabízí celou řadu pomocných tříd. Co mu lze vytknout je možná zbytečnost některých těchto tříd, jelikož jejich účelu lze dosáhnout pomocí grid systému (např. třídy pro zarovnávání). Vzhledem k odlišnosti jednotlivých webových stránek pak nepovažuje autor za vhodné pomocné třídy pro barvy rámečku či pozadí – můžou se hodit pro web spoléhající pouze na samotnou knihovnu, ale nepomohou při vývoji dle zadaného grafického návrhu. [12]

2.2. Foundation

V roce 2008 vytvořila společnost ZURB (https://zurb.com), která se zabývá vývojem softwaru, nástroj pro zjednodušení své práce v jazycích HTML, CSS a JavaScript. Nazvala jej ZURB Style Guide. Tento nástroj byl dále rozvíjen a v roce 2010 byl přejmenován na Foundation. Od roku 2011 je pak dostupný open-source pod licencí MIT. Aktuální verze je Foundation 6. Stejně jako Boostrap je vyvíjen v preprocesoru SASS a využívá taktéž javascriptovou knihovnu jQuery. [19]

Pro resetování některých výchozích stylů používá Foundation Normalize.css (https://necolas.github.io/normalize.css/), což je soubor CSS stylů, který zařizuje vykreslování některých HTML prvků tak, aby se napříč prohlížeči zobrazovaly konzistentněji. Pro rozměry využívá zejména jednotky `rem`, ale například breakpointy grid systému jsou definovány pomocí jednotky `em`, která vychází z velikosti písma rodičovského prvku, tj. pokud má rodičovský prvek velikost písma nastavenu na 10`px`, pak 1`em` = 10`px`. V definici breakpointů pomocí Media Queries se však tato jednotka chová stejně jako `rem`, tj. orientuje se podle velikosti písma prvku `<html>`. Vývojáři Foundation se rozhodli pro použití jednotek `em` v tomto případě zřejmě proto, že použití jednotek `rem` v Media Queries způsobovalo problémy na webových prohlížečích Safari (na nejnovějších verzích toto ale bylo opraveno). Používá taktéž systémové fonty. [19]

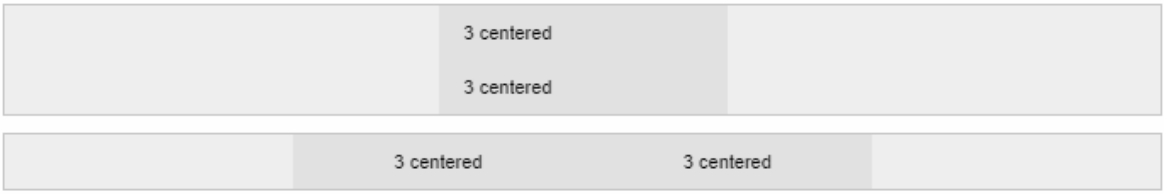
V rámci používání Foundation si vývojář může vybrat (alespoň u některých komponent a grid systémů), zdali chce využívat flexbox nebo ne. Tato volitelná možnost je zde zejména kvůli kompatibilitě napříč prohlížeči, která není stoprocentní, autor práce je však toho názoru, že je dnes již bezpečné jej využívat. Například na stránce https://caniuse.com/#search=flexbox lze vidět, že z prohlížečů, které jsou používány alespoň 0,5 % uživatelů, má problém s flexboxem pouze prohlížeč Internet Explorer a problémy s ním se dají vyřešit. S tím souvisí i to, že Foundation nabízí celkově tři různé grid systémy. [19]

Pokud se vývojář rozhodne flexbox nepoužívat, může tak použít Float Grid, grid systém využívající CSS vlastností `float` a procentuálních šířek. Defaultně má, stejně jako všechny grid systémy knihovny Foundation, 12 sloupců s možností změny. Zápis uvažovaného příkladu z obrázků 3-5 bude vypadat následně:

```
<div class="row">
  <div class="columns small-12 medium-6 large-8">
  </div>
  <div class="columns small-12 medium-6 large-4">
  </div>
</div>
```

Kód 6 – Zápis uvažovaného příkladu z obrázků 3-5 pomocí Float Grid/Flex Grid systémů knihovny Foundation [zdroj autor]

Defaultně má Foundation nastaveny pouze dva breakpointy, přepočítány z jednotek `em` vychází většinou na 640 a 1024 pixelů. Více breakpointů lze sice nastavit, ale i tak by se autor práce přikláněl k více i ve výchozím nastavení. Oproti Bootstrapu si zde v kódu 3 všimnout, že zde není prvek se třídou `container`. Šířku obsahu totiž kontroluje přímo třída `row` – nabízí se tak otázka, zdali by nebylo lepší vlastnosti pro definování grid systému a šířky obsahu rozdělit do více tříd. To se autorovi práce nezamlouvá, nicméně je spokojen se zápisem tříd pro sloupce, kdy se definuje základní třída pro sloupec (v tomto případě columns) a následně se pomocí dalších tříd definuje chování v rámci breakpointů. Float Grid nabízí taktéž třídy pro odsazování prvků, změnu pořadí a centrování. Ohledně odsazování prvků, třídy nejsou tolik užitečné, jelikož většina případů by se i v tomto grid systému dala řešit pomocí prázdných sloupců či centrování. Třídy pro změny pořadí jsou užitečnější, využívají vlastností `float` a pozicování, v případě složitějších případů (kdy chceme měnit pořadí tří prvků a více), je pomocí nich dosáhnout kýženého cíle velice nesnadné. To stejné platí u centrování sloupců. Centrovat je pomocí obsažených tříd lze, ale centrované sloupce se zobrazují automaticky pod sebou (Obrázek 9). [19]



Obrázek 9 - Centrování více prvků v řádku v rámci grid systému. Ve vrchní části je vidět centrování Float Grid systému knihovny Foundation. V dolní části je takové centrování, které by autor práce preferoval. [19] [zdroj autor]

Pokud se vývojář rozhodne flexbox využít, může si vybrat ze dvou grid systémů, které jej používají. Jedná se o *Flex Grid* a *XY Grid*. První z jmenovaných je vlastně jen již popsaný Float Grid vylepšený o vlastnosti flexboxu. I jeho zápis vypadá na první pohled zcela stejně (Kód 6). Taktéž by se dalo

říci, že nabízí obdobné možnosti jako grid systém knihovny Bootstrap. Není třeba tak dál vysvětlovat jeho vlastnosti. [19]

Nejzajímavějším grid systémem Foundation (a zároveň výchozím) je tak právě XY Grid. Nejprve je vhodné ukázat zápis námi uvažovaného příkladu:

```
<div class="grid-container">
  <div class="grid-x grid-padding-x">
    <div class="cell small-12 medium-6 large-8">
    </div>
    <div class="cell small-12 medium-6 large-2">
    </div>
  </div>
<div>
```

Kód 7 - Zápis uvažovaného příkladu pomocí grid systému XY Grid knihovny Foundation [zdroj autor]

Jak lze vidět, zápis je poněkud složitější, než v případě systému Float Grid (resp. i Flex Grid). Třída `grid-container` určuje šířku obsahu. Třída `grid-x` je rodičovský prvek pro jednotlivé sloupce. X na konci třídy značí, že se sloupce zarovnávají vedle sebe. Y by znamenalo, že se řadí pod sebe (tzv. vertikální grid systém). Následná třída `grid-padding-x` udává sloupcům, že se mezi nimi mají tvořit mezery (což je u většiny grid systémů defaultně přednastaveno, např. u zbylých grid systémů Foundation je to 0,9375`rem` ≈ 30 pixelů) a v jakém směru (x/y). Obdobně jsou obsaženy i třídy `grid-margin-x` (a `grid-margin-y`). Vývojář si tak může určit, zdali mezery budou vytvořeny pomoci CSS vlastnosti `margin` (vnější okraje prvků) či `padding` (vnitřní okraje prvků). Následují pak definice samotných sloupců, kde se na rozdíl od zbylých Foundation grid systémů nazývá definující buňka `cell`, nikoliv `columns`. Kromě definicí velikostí lze používat i třídy `shrink` a `auto`. První definuje sloupec široký pouze pro potřebu jeho obsahu a druhá sloupce, které se rovnoměrně rozprostřou do zbývajícího prostoru řádku (Obrázek 10). [19]

Obrázek 10 - Ukázka sloupců s třídami shrink a auto. Sloupec s textem Shrink má šířku postačujícím rozměrům jeho obsahu a druhý sloupec se rozprostřel do zbytku prostoru řádku [19]

Samozřejmě lze využívat i flexboxové třídy pro horizontální i vertikální zarovnávání prvků či změnu pořadí. Poslední vlastností XY Grid systému je třída `grid-frame`, čerpající z příbuzného frameworku Foundation for Apps (https://foundation.zurb.com/apps.html). Slouží převážně k vytváření celkového rozložení stránky pomocí flexboxu. Celkově je tak XY Grid jedním z nejkomplexnějších grid systémů vůbec (nebo co se alespoň týče systémů popsaných v této práci), dle autora práce je však příliš složitý a všechny jeho možnosti by sám zřejmě nevyužil. [19]

Co se týče již hotových komponent, rozděluje je Foundation do několika kategorií – controls (ovládací prvky), navigations (menu, stránkování, drobečková navigace aj.), containers (rodičovské prvky dalšího obsahu, např. rozklikávací záložky, karty, modální okna, tabulky) a media (štítky, videa, progress bary). Oproti Bootstrapu je jejich defaultní grafický styl více odlehčený, což může být užitečné zejména v případě, že hotovou komponentu chceme použít při vývoji dle různorodého grafického návrhu.

Pomocné třídy jsou pak v knihovně velmi důležité, zejména pak ty týkající se vlastností flexboxu, jelikož se využívají i v rámci grid systémů. Jinak zahrnují většinu vlastností CSS a pro velkou část z nich existují i jejich responzivní ekvivalenty (tj. s prefixem dle velikosti obrazovky, small-, medium-, large-), což ale poněkud zvětšuje velikost výsledného CSS souboru.

2.3. Pure CSS

Pure CSS je knihovna vznikající od roku 2013. Již defaultně je tvořena jako skupina několika balíků tříd – base, grids, forms, buttons, tables, menus. Její největší výhoda je její opravdu malá velikost výsledného CSS souboru. Je vyvíjena v samotném jazyku CSS a nepoužívá, na rozdíl od větších knihoven JavaScript (ani jQuery), jelikož neobsahuje dynamické komponenty, které by jej nutně potřebovaly (např. *carousel* či modální okna). [20]

Pro resetování stylů využívá knihovna, stejně jako Foundation, známý soubor Normalize.css a taktéž systémové fonty. Pro většinu velikostí používá pak jednotky `em`, případně `px` (pro definici breakpointů, některých horizontálních mezer aj.). [20]

Grid systém, obsažen v balíku grids, má v základním nastavení definované 4 breakpointy (sm 568 pixelů, md 768 pixelů, lg 1024 pixelů a xl 1280 pixelů) a je mobile-first. Zápis uvažovaného příkladu z obrázků 3-5 v něm vypadá takto:

```
<div class="pure-g">
  <div class="pure-u-5 pure-u-md-12-24 pure-u-lg-18-24">
  </div>
  <div class="pure-u-5 pure-u-md-12-24 pure-u-lg-6-24">
  </div>
</div>
```

Kód 8 – Zápis uvažovaného příkladu z obrázků 3-5 v grid systému knihovny Pure CSS [zdroj autor]

Třída `pure-g` je rodičovský prvek pro jednotlivé sloupce, na rozdíl od jiných systémů neurčuje šířku obsahu (taková třída v Pure CSS není). Jednotlivé sloupce určují třídy s prefixem `pure-u`. Následně se dvěma písmeny definuje breakpoint (či se vynechává v případě definice sloupců

pod 568 pixelů) a poté dvě čísla. První znamená počet sloupců, které prvek bude zabírat a druhé z kolika sloupců. Pure CSS totiž nabízí naráz třídy pro pětisloupcový i čtyřiadvacetisloupcový grid systém. Tyto třídy se dají naráz kombinovat u jednoho prvku. Na rozdíl od jiných grid systémů nenabízí žádné pomocné třídy pro zarovnávání obsahu, odsazování, změnu pořadí sloupců atd., z nichž některé by autorovi práce podstatně chyběly. [20]

Hotové komponenty lze nalézt v balících *forms*, *buttons*, *tables* a *menus*. Z jejich názvu lze vyčíst, jaké komponenty obsahují. Jejich vzhled lze snadno upravit, ale bohužel zde nelze nalézt složitější komponenty – k nim by totiž zřejmě bylo potřeba použít JavaScript, který v knihovně není obsažen. Z pomocných tříd obsahuje Pure CSS pouze třídy pro skrývání prvků a zobrazování obrázků. [20]

2.4. Bulma

Bulma je CSS knihovna vyvíjecí se jako open-source pod licencí MIT od roku 2016. Jedná se tak, vzhledem k vzrůstající popularitě této knihovny, o jednu z nejrychleji se rozvíjejících knihoven. Pro srovnání například Foundation, dostupné veřejně od roku 2011, má k dnešnímu datu (24. 2. 2018) na serveru Github (https://github.com) přidáno v oblíbených cca 27 000 uživatelů a Bulma je na tom podobně – skoro 25 000. Knihovna je vyvíjena v preprocesoru SASS. [19] [21]

Pro resetování některých výchozích stylů používá Bulma příbuzný soubor `minireset.css`. Jeho autorem je vlastně autor samotné knihovny Bulma, Jeremy Thomas. Používá v něm, stejně jako naprostá většina ostatních knihoven, systémové fonty. Jako jednotky využívá knihovna dle potřeby jak `em`, tak `rem` i pixely. [21]

Grid systém knihovny Bulma má v základním nastavení 5 breakpointů – tablet 769`px`, desktop 1024`px`, widescreen 1216`px` a FullHD 1408`px`. Zápis uvažovaného příkladu z obrázků 3-5 vypadá pak takto:

```
<div class="container">
  <div class="columns">
    <div class="column is-12-mobile is-6-tablet is-8-desktop">
    </div>¨
    <div class="column is-12-mobile is-6-tablet is-4-desktop">
    </div>
  </div>
</div>
```

Kód 9 - Zápis uvažovaného příkladu z obrázků 3-5 v grid systému knihovny Bulma [zdroj autor]

Třída `container` určuje šířku obsahu, `columns` je rodičovská třída pro následné sloupce. Jejich třídy připomínají poměrně třídy knihovny Foundation. Kromě tříd ve tvaru `is-(číslo)-`

`(breakpoint)` nabízí Bulma třídy jako např. `is-three-quaters-(breakpoint)`, `is-two-thirds-(breakpoint)`, `is-half-(breakpoint)` a podobně. Jedná se o třídy definující anglicky poměr, jaký bude sloupec v řádku zabírat. Autorovi práce tyto třídy přijdou zbytečné – stačily by dle něj třídy ve tvaru ukázaném v kódu 9. Bez těchto tvarů pak třída column funguje tak, že si rozdělí zbylý prostor v řádku (Obrázek 11). [21]

Obrázek 11 - Ukázka chování tříd column. První sloupec zabírá tři čtvrtiny místa řádku. Další sloupce bez tříd definující velikost si zbylý obsah řádku rovnoměrně rozdělí. [21]

Stejně tak lze sloupci přidat třídu `is-narrow`, která funguje stejně jako třída `shrink` v knihovně Foundation (viz Obrázek 10). Mimo to grid systém nabízí opět třídy pro centrování a odsazování prvků. Zvláštní je, že v samotné dokumentaci je psáno, že odsazování sloupců může být řešeno použitím prázdných sloupců, avšak stejně jsou třídy odsazení obsaženy. Použití tříd je sice pro vývojáře pohodlnější, avšak navyšuje velikost výsledného CSS souboru. Mimo, dá se říci tradiční, grid systém, nabízí Bulma i tzv. tile grid systém, který se skládá z dlaždic, které je schopen distribuovat jak v horizontálním, tak ve vertikálním směru. Příklad, který demonstruje, co lze s jeho pomocí vytvořit, může vypadat následně:

Obrázek 12 - Příklad vytvořený pomocí tile grid systému knihovny Bulma [21]

Z hotových komponent obsahuje Bulma menu, patičku, formulářové prvky, různé karty, tlačítka, progress bary, tabulky, modální okna, rozklikávací záložky i textové prvky. Pojem hotové komponenty v tomto případě ale není úplně přesný. Bulma neobsahuje žádné soubory JavaScriptu, které by v normálním případě byly ke správnému fungování některých komponent (například

modálního okna) nutné. Komponenty jsou tak hotové pouze co se týče vzhledové stránky, takže některé z nich nemůže použít bez dalších úprav či dodělávek. Pokud vývojář tak vyvíjí stránku dle grafického návrhu, nejsou mu tak komponenty tolik užitečné, jelikož by ocenil spíše hotové chování komponent a ne vzhled. [21]

Bulma obsahuje pomocné třídy pro základní vzhledové vlastnosti některých komponent, zarovnávání i pozicování, zobrazování i textové vlastnosti. Třídy jsou poměrně užitečné, jelikož pokrývají nejčastější vlastnosti, které vývojáři používají. Jediné, co by autor práce vytknul, jsou třídy určující barvu textů, tlačítek a dalších komponent. Často se nestane, že by je vývojář použil v případě vývoje dle grafického návrhu a přesto, že je lze v SASS souborech měnit, stále by tyto vlastnosti raději vynechal, takového názoru je alespoň autor práce. [21]

2.5. Závěry z provedené analýzy

Autor v předešlých kapitolách provedl analýzu čtyř stávajících knihoven z hlediska použitých technologií, grid systému, hotových komponent a pomocných tříd. V této kapitole by rád uvedl, jak jednotlivá hlediska ovlivní vývoj jím vytvořené knihovny. Je jasné, že se knihovnami, které měl tu možnost analyzovat, mohl nechat v jednotlivých částech inspirovat.

Použité technologie

Nejlepším řešením bude dle autora knihovnu vyvinout v preprocesoru SASS. Preprocesor umožňuje knihovnu vyvinout rychleji a dává možnost případným vývojářům knihovnu lépe upravit svým požadavkům díky proměnným a funkci import, pomocí které snadno může vývojář určit, jaké části knihovny (ne)použije. I přes existenci více preprocesorů bude použit SASS, zejména proto, že většina projektů ve firmě Appio jej využívá. Pro správné fungování knihovny pak bude nutno použít jazyk JavaScript (případně jeho knihovny), jelikož autor chce, aby vytvořené komponenty byly plně funkční, jak tomu je u knihoven Bootstrap a Foundation. [12] [19]

Pro resetování některých stylů nepoužije autor žádný již vytvořený soubor, ale případně se nechá inspirovat jednotlivými CSS pravidly, které obsahují. Nebude taktéž udávat základní vzhled (jako barvu, velikost písma apod.) textovým prvkům, bude jim však možno přiřadit některé třídy komponent. Nerad by zahrnoval resetování stylů pro prvky, u kterých není předpokládáno použití tříd samotné knihovny – resetovány budou pouze styly prvků nadpisů, odstavce, seznamů, textových polí, tlačítek a odkazů. To ušetří velikost výsledného CSS souboru. Je možné, že s postupným vývojem knihovny a použitím na reálných projektech se však výsledné styly resetování budou upravovat a zvětšovat dle potřeby vývojářů. Defaultně, stejně jako všechny analyzované knihovny,

použije nativní fonty operačních systémů, jelikož nechce nutit vývojářům načítat určitý font. Fonty se navíc ve vyvíjených projektech často liší, a tak by to nebylo vhodné. Pro rozměry budou použity jednotky `rem`, pro definice breakpointů pak `em`, stejně jako u Foundation. [12]

Grid systém

Grid systém by měl obsahovat třídu, která bude definovat šířku obsahu a následnou rodičovskou třídu, ve které budou definovány jednotlivé sloupce. Zápis sloupců se bude inspirovat knihovnou Foundation, tj. bude zde základní třída definující jednotlivé sloupce a dalšími třídami se bude deklarovat další chování. Grid bude defaultně dvanáctisloupcový s možností úpravy a bude využívat flexbox. Samozřejmostí tak bude obdoba tříd shrink a auto z knihoven Foundation / Bulma. Zápis breakpointů ve třídách se však bude inspirovat Bootstrapem, jelikož na jeho zápis breakpointů je skupina vývojářů ve firmě Appio zvyklá. Grid systém bude obsahovat pomocné třídy vlastností flexboxu, ale nebude obsahovat třídy pro odsazování prvků a další, dle autora nepotřebné třídy zmíněné v předešlých kapitolách 2.1 a 2.2. [19] [21]

Hotové komponenty

Z hlediska hotových komponent se autor příliš hotovými řešeními inspirovat nenechá, spíše bude vycházet z již vytvořených projektů. Obsahově by však nechtěl vytvářet přespříliš velké množství komponent, ale zároveň by jich mělo být více než v analyzované knihovně Pure CSS. Právě u komponent bude muset deklarovat prvkům i některý základní vzhled. Měl by být však co nejvíce decentní, aby jej šlo rychle upravit. To se dle jeho názoru například Bootstrapu v některých případech nedaří.

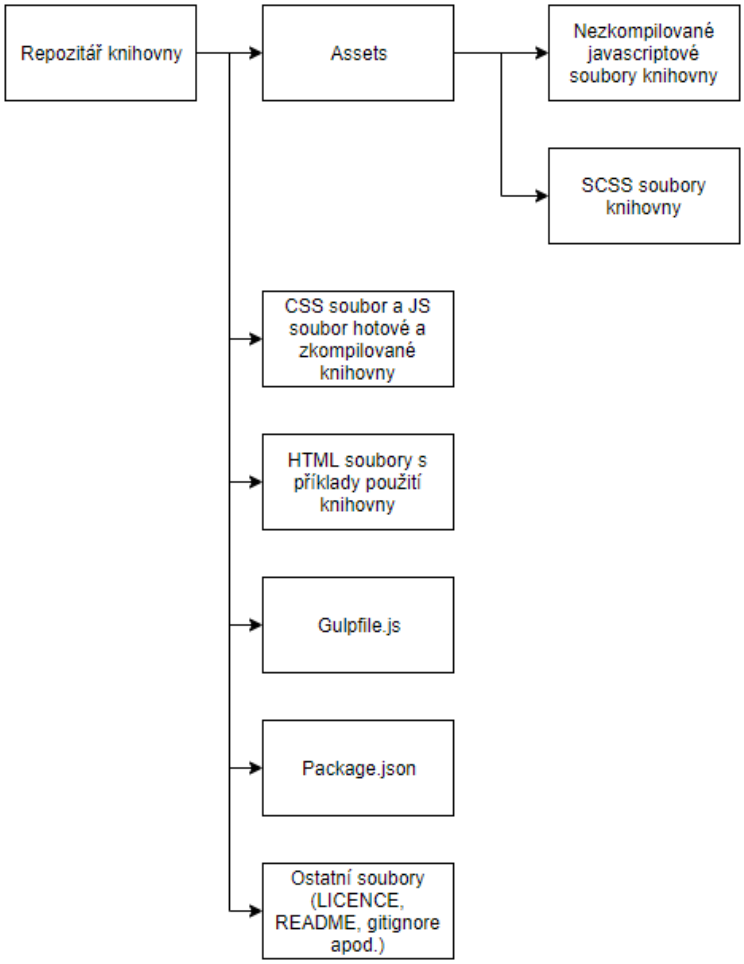
Pomocné třídy

U pomocných tříd bude nutno zhodnotit, zdali je vlastnost, kterou dané třídy budou pokrývat, natolik používaná, že ji má smysl zahrnovat do knihovny. Autor při vývoji zřejmě projde i pomocné třídy analyzovaných knihoven, aby posoudil, zda obdobné třídy nezahrne do knihovny jím vytvářené.

3. Struktura a vize knihovny

Tato kapitola má za úkol popsat plánovanou strukturu knihovny a její vizi.

Repozitář knihovny se bude skládat ze složky *assets*, kde bude knihovna dostupná ve formátu před kompilací (tj. v preprocesoru SASS a v neminifikovaném JavaScriptu), složky *dist* se zkompilovaným CSS a JS souborem samotné knihovny a ze složky *examples*, která bude obsahovat HTML soubory se příklady použití jednotlivých částí knihovny. Mimo zmíněných složek jsou důležité ještě soubory `gulpfile.js` a `package.json`, kvůli samotné kompilaci knihovny. Strukturu znázorňuje Obrázek 13.



Obrázek 13 - Schéma struktury knihovny [zdroj autor]

Co na schématu najít nelze, je výčet komponent, které knihovna bude obsahovat. Ty zahrnují některé textové prvky, responzivní obrázky, videa a iframe prvky, karty, horizontální menu, vertikální menu,

carousel, patičku, vizuály, formulářové prvky, drobečkovou navigaci, stránkování, progress bar, modální okno a rozklikávatelné záložky. Mimo ně samozřejmě bude knihovna obsahovat i resetující styly, pomocné třídy a grid systém.

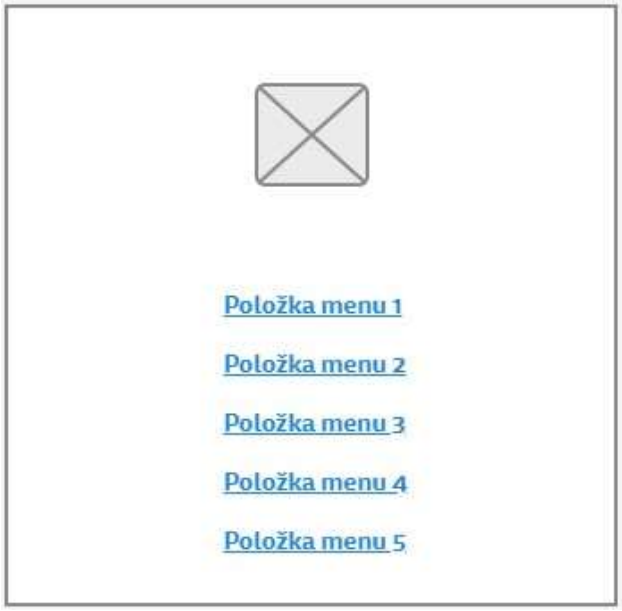
Pro následující komponenty bude knihovna využívat i JavaScript (resp. jQuery):

- Horizontální menu
- Vertikální menu
- *Carousel*
- Modální okno
- Záložky
- Media

Pro složitější komponenty, konkrétně menu, modální okno a záložky byly před jejich vývojem vytvořeny i *wireframy*, tj. zjednodušené skici webu, struktury, které popisují funkcionalitu a rozmístění prvků na stránce. U komponent menu byly vytvořeny 3 wireframy, jeden pro mobilní zobrazení a dva pro klasické zobrazení horizontálního a vertikálního menu. Všechny wireframy byly vytvořeny pomocí webové služby Mockingbird (https://gomockingbird.com/). [22]

Obrázek 14 - Wireframe mobilního zobrazení menu (nalevo před otevřením a napravo po otevření menu kliknutím na tzv. hamburger ikonu vpravo nahoře). Takovým menu se říká hamburger menu. [zdroj autor]

Obrázek 15 - Wireframe klasického zobrazení horizontálního menu [zdroj autor]



Obrázek 16 - Wireframe klasického zobrazení vertikálního menu [zdroj autor]

Obrázek 17 - Wiframe komponenty záložek [zdroj autor]

Obrázek 18 - Wireframe komponenty modálního okna [zdroj autor]

Z hlediska kompatibility se autor bude zabývat prohlížeči, které využívá více než 0,5 % uživatelů dle serveru caniuse.com k březnu 2018 ([8]). Jedná se tak o verze desktopových prohlížečů Internet Explorer 11, Edge 16+, Mozilla Firefox 57+, Google Chrome 49+, Safari 11+ a verze mobilních prohlížečů Safari 10.2+, Opera Mini, Android Browser 4.4+, Google Chrome 64+, UC Browser for Android 11.8+ a Samsung Internet 4+. Knihovna samozřejmě může fungovat i na jiných prohlížečích, nicméně autor práce to nebude testovat. Je také možné, že některé vlastnosti nebudou fungovat ve všech prohlížečích, to však autor případně zmíní v dokumentaci knihovny a navrhne alternativní řešení.

Sama knihovna bude vyvíjena pod názvem *skar-is* (zkratka pro *Skara Interface Solution*) a bude umístěna na adrese https://github.com/skaryys/skar-is. Knihovna tak bude dostupná open-source pod licencí MIT (https://opensource.org/licenses/MIT). Vzhledem k tomu, že autor k ní zamýšlí napsat i dokumentaci, ve které by měly být obsáhlé HTML příklady, bude taková dokumentace dostupná na zvláštní webové stránce na doméně http://skaris.skaramart.in. Knihovna tak bude dostupná ze tří zdrojů – této domény, GitHub repozitáře a taktéž ji bude možno nainstalovat pomocí balíčkovacích systémů Yarn a NPM.

4. Vývoj knihovny

Po analýze existujících řešení a určení struktury, vize a technologií, kterých bude knihovna využívat, lze přistoupit k samotné realizaci knihovny. Tato kapitola je tak věnována celému vývoji jednotlivých částí knihovny.

Před samotným vytvořením SCSS a JS částí knihovny je zapotřebí určit i nároky knihovny na straně HTML kódu. Typická HTML šablona se vším, co je potřeba ke správnému fungování knihovny pak bude vypadat následně:

```
<!DOCTYPE html>
<html>
    <head>

        <!-- recommended meta tags -->
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-
scale=1, maximum-scale=1, shrink-to-fit=no">

        <!-- skar-is CSS -->
        <link rel="stylesheet" href="main.css">
        <title>Title</title>

    </head>
    <body>

        <!-- webpage content here -->

        <!-- jQuery and skar-is JS-->
        <script src="http://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha256-3edrmyuQ0w65f8gfBsqowzjJe2iM6n0nKciPUp8y+7E="
crossorigin="anonymous"></script>
        <script src="main.js"></script>

    </body>
</html>
```

Kód 10 - HTML šablona pro správné použití knihovny [zdroj autor]

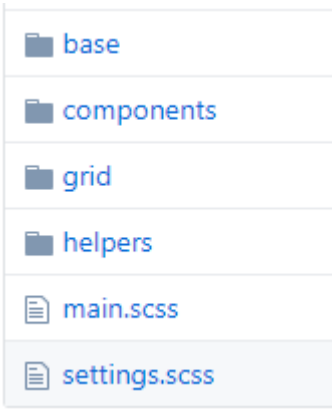
Lze si všimnout, že pro správné fungování je potřeba pouze knihovna jQuery (pokud je použita i javascriptová část knihovny) a meta tagy `charset="utf-8"`[3], `http-equiv="X-UA-`

[3] Definuje kódování textu na stránce

`Compatible"`[4] a `viewport`[5]. Meta tagy mohou být i pozměněny, nejedná se o závazný zápis. Knihovna by měla fungovat i pod jinou definovanou znakovou sadou pomocí meta tagu `charset` a taktéž některé části meta tagu `viewport` mohou být změněny, například v případě, že je pomocí knihovny vyvíjena neresponzivní webová stránka. Za nejdůležitější tak lze považovat tag `http-equiv="X-UA-Compatible"`, který zajišťuje lepší (modernější) chování prohlížeče Internet Explorer.

4.1. Základy knihovny

Základem SCSS části knihovny jsou soubory `main.scss` a `settings.scss`. `Main.scss` do sebe importuje všechny ostatní SCSS soubory a `settings.scss` obsahuje proměnné, které jsou využívány všemi ostatními částmi knihovny. V případě potřebného modifikování knihovny je tak možné základní vlastnosti měnit pouze změnou tohoto souboru.



Obrázek 19 - Struktura SCSS části knihovny [zdroj autor]

Povinnými částmi pro další fungování knihovny jsou pak ještě soubory obsažené ve složce *base*. Jedná se o funkce, mixiny a resetující styly. Funkce a mixiny využívají další části knihovny, resetující styly pak definují vlastnosti HTML elementů, jejichž chování se napříč webovými prohlížeči liší nebo jejichž výchozí chování nevyhovuje dalším částem knihovny. V souboru s resetujícími styly ale lze najít i definice základních globálních stylů, kterými mohou být např. barva pozadí celé stránky či výchozí velikost písma – ta je pro kořenový element `<html>` nastavena na 62,5% velikosti písma

[4] Definuje v jakém režimu se stránka vykreslí v prohlížeči Internet Explorer

[5] Pomocí meta tagu `viewport` se nastavuje rozlišení celého webu (pokud je responzivní tak se rozlišení rovná šířce zařízení), měřítko v jakém jej zobrazovat, maximální přibližování obsahu webu, zakázání přibližování obsahu webu a případně další specifické chování s tím související pro některá zařízení (například iPhone X).

prohlížeče, což ve většině případů vychází na 10 pixelů. Pro element těla stránky (`<body>`) je pak ale velikost písma nastavena na 1,6`rem`, tj. přepočtem 16 pixelů. A takto je to s většinou rozměrů v celé knihovně, které jsou definovány právě v jednotkách `rem`, 1`rem` se tak v rámci knihovny rovná při typickém nastavení prohlížeče deseti pixelům. Pokud má uživatel nastavenu jinou velikost písma prohlížeče, rozměry se tak mění dle ní. Díky tomu nebude výsledná stránka rozhozená ani při nevšedním nastavení písma prohlížeče.

Z proměnných knihovny jsou nejdůležitější pak definice breakpointů[6], s nimiž pracuje jak grid systém knihovny, tak některé komponenty. Na rozdíl od většiny ostatních rozměrů jsou definovány v jednotkách `em`, jelikož některé prohlížeče (zejména Safari) mohou mít s `rem` breakpointy problémy. Knihovna má definovaných 5 breakpointů – xs 30`em`, sm 48`em`, md 64`em`, lg 75`em` a xl 82.5`em`. V přepočtu na pixely se defaultně jedná o 480`px`, 768`px`, 1024`px`, 1200`px` a 1320`px`. Breakpointy mohou být měněny, důležité je však, že dle nich se orientuje i šířka kontejneru (a tudíž i šířka celého obsahu stránky). Např. na rozlišeních 1200`px` – 1319`px` má při klasické velikost písma kontejner šířku 1200 pixelů. V případě, že vývojář kontejner nevyužije, může s breakpointy pracovat i tak. Pomocí knihovny mohou být vytvářeny i stránky, jejichž obsah zabírá 100 % šířky okna prohlížeče, jen je pak potřeba nepoužívat defaultní třídy `.container` z grid systému knihovny.

Poslední důležitou informací o základech knihovny jsou použité systémové fonty. Aby byl i v defaultním nastavení zachován hezký vzhled písma, vypadá pak definice proměnné `$fontFamily`, která určuje font pro celou webovou stránku, takto:

```
$fontFamily: -apple-system, system-ui, "Segoe UI", Roboto, Oxygen,
"Fira Sans", "Droid Sans", Ubuntu, Cantarell, sans-serif;
```

Kód 11 - Definice proměnné $fontFamily určující font pro celou webovou stránku [zdroj autor]

Počet definovaných fontů se může zdát větší, jelikož pro různé operační systémy je nutno definovat jiné fonty. Výše uvedený zápis se stará o definice fontů na operačních systémech iOS, macOS, Windows, Android, Chrome OS, Firefox OS a na některých linuxových distribucích.

[6] Body rozlišení, ve kterých se změní chování či styly určitých prvků na stránce. Viz kapitola 1.1.3.

4.2. Pomocné třídy

SCSS část knihovny obsahuje taktéž 5 typů pomocných tříd. Jedná se o třídy ovlivňující zobrazování prvků, obtékání prvků, pozicování, zarovnávání a zalamování textu a třídy pro rychlé vytvoření hover efektu (efekt po najetí myší na prvek) odkazů.

```
.h-linkUnderlineHover {
  text-decoration: none !important;
  &:hover {
    text-decoration: underline !important;
  }
}
```

Kód 12 - Pomocná třída pro rychlé vytvoření hover efektu odkazu v SCSS [zdroj autor]

V kódu 12 lze vidět zápis pomocné třídy pro rychlé vytvoření hover efektu odkazu. Lze vidět, že u jednotlivých vlastností třídy je ještě zápis `!important`. Ten je použit u všech vlastností pomocných tříd. Zajišťuje tak, že vlastnosti pomocných tříd jsou uplatňovány přednostně – tudíž styly pomocných tříd by měly být aplikovány vždy, alespoň pokud není knihovna upravena či nejsou aplikovány další vlastní styly vývojáře.

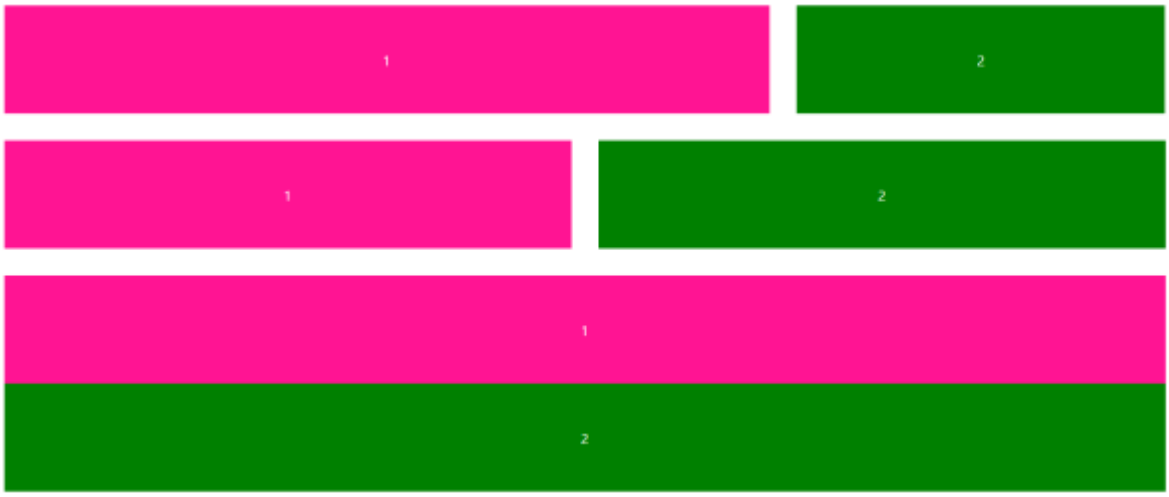
Některé pomocné třídy využívají i vlastnosti, které nejsou podporovány všemi prohlížeči, na které je knihovna uzpůsobena. Na to je však potenciální vývojář upozorněn v dokumentaci knihovny.

4.3. Grid systém

Grid systém vytvořené knihovny tvoří třída `container`, která definuje rozměry obsahu stránky a zarovnává jej na střed. V rámci ní lze použít třídu `row`, což je rodičovský prvek, v němž jsou následně definovány jednotlivé sloupce. Zápis příkladu z obrázků 3-5 pomocí vytvořené knihovny je ukázán v následujícím kódu a výsledek v následujícím obrázku.

```
<div class="container">
  <div class="row>
    <div class="column xs-12 sm-6 lg-8">
      1
    </div>
    <div class="column xs-12 sm-6 lg-4">
      2
    </div>
  </div>
</div>
```

Kód 13 - Zápis uvažovaného příkladu z obrázků 3-5 pomocí vytvořené knihovny [zdroj autor]



Obrázek 20 - Ukázka uvažovaného příkladu z obrázků 3-5 vytvořená pomocí knihovny skar-is. První řádek představuje zobrazení prvků na desktopovém rozlišení, druhý na tabletu a poslední na mobilním telefonu. [zdroj autor]

Samotný sloupec definuje třída `column` – bez použití dalších tříd se s ní pak sloupce chovají tak, že si rovnoměrně rozdělí zbývající prostor řádku. V případě použití tříd ve tvaru `(breakpoint)-(číslo)` pak sloupce zabírají určitý počet sloupců s celkově možného počtu sloupců (defaultně 12). Další možností je použití tříd ve tvaru `(breakpoint)-stretch` nebo `(breakpoint)-default`. Sloupec s první ze zmíněných tříd bude široký pouze tak, jak široký je jeho obsah a druhá zmíněná třída navozuje opět výchozí chování sloupce v případě, že na menších rozlišeních se sloupec chová jinak (nedefaultně).

Grid systém má však kromě, dalo by se říci základního chování, více možností než jen měnit rozměrové chování sloupců. Celkově je rozdělen do devíti částí, z nichž první je de facto vysvětlena v odstavci výše. Mezi další části patří:

- **Orders** – třídy pro změnu pořadí sloupce. Sloupec lze rámci breakpointů ovlivňovat tak, aby byl první či poslední.
- **Align-self** – třídy pro změnu flexbox vlastnosti `align-self` jednotlivých sloupců (sloupec je v kontextu k `row` vlastně flex položkou)
- **Directions** – ovlivňuje vlastnost `flex-direction` třídy `row`
- **Wraps** – ovlivňuje vlastnost `flex-wrap` třídy `row`
- **Justify** – ovlivňuje vlastnost `justify-content` třídy `row`
- **Items** – ovlivňuje vlastnost `align-items` třídy `row`
- **Content** – ovlivňuje vlastnost `align-content` třídy `row`

- **Hide –** umožňuje prvky skrývat a zase zobrazovat v rámci breakpointů

Jak je ze seznamu vidět, většina tříd ovlivňuje flexboxové vlastnosti – autor práce se snaží těchto vlastností co nejvíce využít. Díky nim lze snadno obsah různorodě zarovnávat a s drobnými úpravami může být tato část knihovny použita i pro tvorbu vertikálního grid systému. Flexboxové třídy grid systému se totiž orientují dle osy flex kontejneru (tím je v tomto případě `row`). V souboru nastavení – `settings.scss` je pak proměnná, ve které jsou jednotlivé názvy částí grid systému. Smazáním některých z nich pak může vývojář určit, které části potřebuje a díky nevytvoření některých částí zmenšit tak velikost výsledného CSS souboru.

Při vývoji grid systému narazil autor na problém s prohlížečem Internet Explorer kvůli jeho špatné podpoře flexboxu. Musel tak k zápisu flexboxových vlastností vždy definovat jednotky a u sloupců grid systému definovat maximální šířku stejnou jako vlastnost `flex-basis`. Tyto opravy zprovoznily grid systém i na prohlížeči Internet Explorer, avšak navýšily velikost výsledného CSS souboru.

4.4. Komponenty

Největší částí knihovny jsou předpřipravené komponenty, kvůli některým z nich knihovna využívá i JavaScript. Každá z komponent je obsažena ve vlastním SCSS souboru, případně má i vlastní javascriptový soubor. Většina komponent obsahuje vlastnosti, které lze snadno změnit v souboru `settings.scss`. V následujících podkapitolách jsou jednotlivé komponenty popsány.

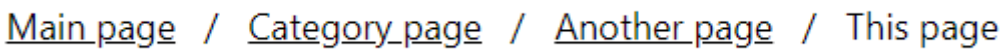
4.4.1. Textové komponenty

Jako textové komponenty je označována sada komponent nadpisů, komponenta odstavce a komponenta seznamu. Tyto komponenty autor zařadil do knihovny spíše jen pro případ, že by pomocí knihovny byl vytvářen jednoduchý web bez dalších externích stylů. Na různých projektech se totiž často textové prvky hodně liší a není tak moc užitečné upravovat již hotové komponenty. Pomocí proměnných je však těmto komponentám možné měnit vnější okraje, tučnost písma, velikost písma a velikost řádkování.

Obrázek 21 - Textové komponenty vytvořené knihovny [zdroj autor]

4.4.2. Drobečková navigace

Další z komponent je hotová drobečková navigace. Jako děliče mezi jednotlivými částmi cesty v navigaci jsou zde defaultně použita lomítka. Pomocí proměnných lze u komponenty měnit mezery a velikost písma.



Obrázek 22 - Komponenta drobečkové navigace vytvořené knihovny [zdroj autor]

4.4.3. Carousel

Pro použití komponenty *Carousel* vytvořené knihovny je zapotřebí použít i javascriptovou část knihovny. Následně lze pak *carousel* ovládat šipkami vpravo a vlevo či přejít na určitý snímek *carouselu* kliknutím na ovládací prvky jednotlivých snímků. Případně lze *carouselu* nastavit možnosti toho, aby snímky po určité době měnil sám HTML atributem `data-autoslide`. HTML zápis *carouselu* se všemi použitými ovládacími prvky může vypadat následně:

```
<div class="c-carousel" data-carousel="carousel-1" data-autoslide="3000">
  <div class="previous-slide" data-previous-carousel="carousel-2"></div>
  <div class="slide">
    <!—obsah snímku -->
  </div>
  <div class="slide">
    <!—obsah snímku -->
  </div>
  <div class="next-slide" data-next-carousel="carousel-2"></div>
  <ul class="indicators">
    <li class="active" data-numbers-carousel="carousel-2" data
number="1"></li>
    <li data-numbers-carousel="carousel-2" data-number="2"></li>
    <li data-numbers-carousel="carousel-2" data-number="3"></li>
    <li data-numbers-carousel="carousel-2" data-number="4"></li>
    <li data-numbers-carousel="carousel-2" data-number="5"></li>
  </ul>
</div>
```

Kód 14- HTML zápis komponenty Carousel vytvořené knihovny [zdroj autor]

Carouselu lze pomocí proměnných nastavit i kolik snímků najednou má zobrazovat a také lze ovlivnit rozměry a barvu ovládacích prvků, pokud jsou použity.

Obrázek 23 - Ukázka komponenty Carousel vytvořené knihovny [zdroj autor]

4.4.4. Patička

Komponenta patičky obstarává tzv. sticky footer, patičku, která zůstává na spodku okna prohlížeče bez ohledu na to, jak vysoký je zbylý obsah stránky. Toto chování je defaultně funkční jenom od breakpointu sm výše, jde však přenastavit pomocí proměnné. Mimo to jde patičce ještě nastavit i výška a barva pozadí.

4.4.5. Formulářové prvky

Knihovna obsahuje několik komponent formulářových prvků. Jednak se jedná o komponentu `c-input`, definující vzhled textových polí a select prvků a jednak o další komponenty, mezi které patří komponenty pro tlačítka, zaškrtávací či výběrová pole (checkbox a radio inputy), popisek pole, chybovou hlášku pole, či komponenta zadávání rozsahu (input typu range). Pro komponentu checkbox a radio inputu je potřeba používat i SVG grafiku. Tu vývojář nalezne v dokumentaci knihovny či může použít vlastní dle libosti. Některé komponenty formulářových prvků se pak mohou drobně lišit na prohlížečích Internet Explorer, Edge a Mozilla Firefox. Týká se to zejména komponenty pro element input typu range. Většina chování formulářových komponent je ale nastavena tak, aby pokud možno na všech prohlížečích bylo dosaženo stejného výsledku. Pomocí proměnných lze měnit jak rozměry formulářových komponent, tak i barvy rámečků, písma a ovládacích prvků (např. šipky u select elementu).

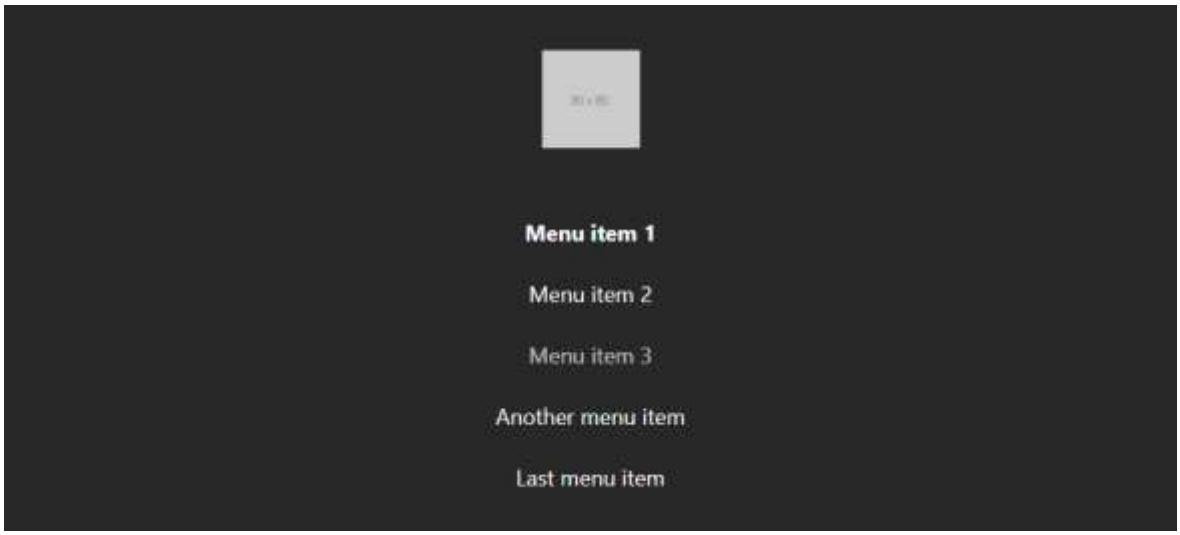
Obrázek 24 - Ukázka formulářových komponent vytvořené knihovny [zdroj autor]

4.4.6. Menu

Komponenta menu představuje responzivní menu. V mobilním zobrazení se jedná o tzv. hamburger menu (viz obrázek Obrázek 25). Dle přidání třídy `horizontal` či `vertical` se následně menu dle nastaveného breakpointu změní na normální horizontální či vertikální menu. Kvůli otevírání responzivního hamburger menu je zapotřebí použít i javascriptovou část knihovny. Pomocí proměnných lze měnit u této komponenty barvy pozadí a písma menu, velikost písma, rozměry menu a breakpoint, kdy se menu změní z hamburger zobrazení na normální.

Obrázek 25 - Hamburger zobrazení menu komponenty vytvořené knihovny [zdroj autor]

Obrázek 26 - Normální zobrazení menu komponenty vytvořené knihovny (horizontální) [zdroj autor]



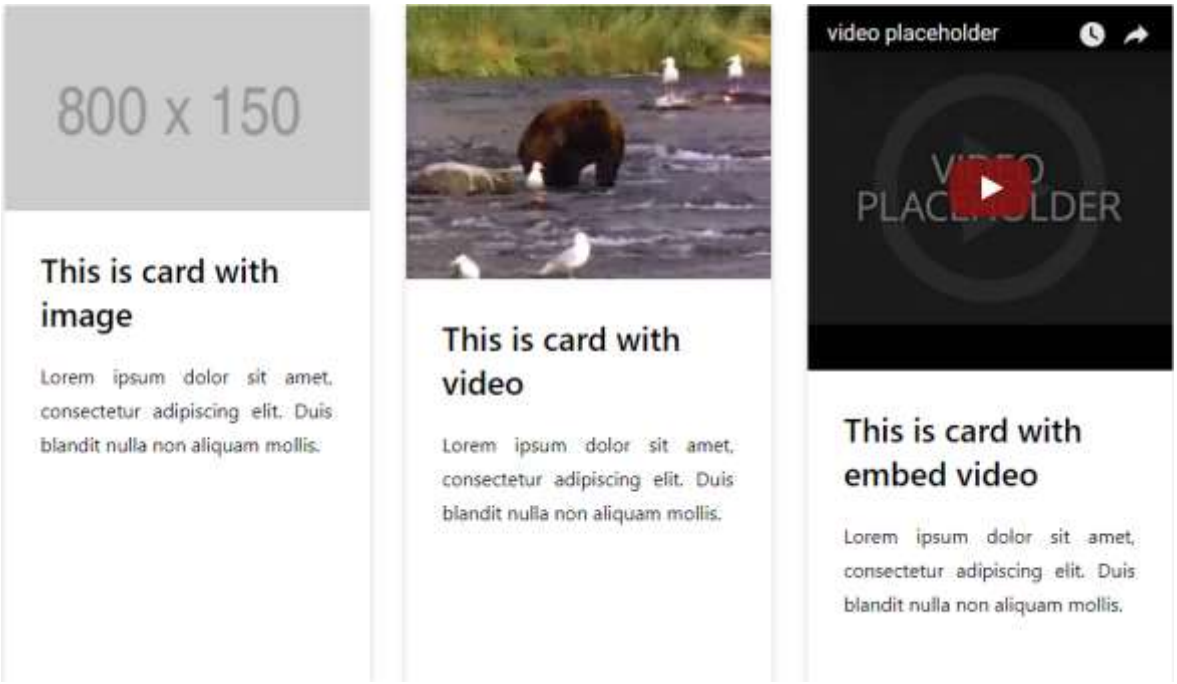
Obrázek 27 - Normální zobrazení komponenty vytvořené knihovny (vertikální) [zdroj autor]

4.4.7. Karta

Komponenta karty slouží pro rychlé vytvoření stejně vypadajících karet, v nichž je následně další obsah. Pomocí proměnných je možné měnit barvu pozadí karet, stín karet či vnitřní okraje obsahu karty. Karty obsahující komponenty Medií a texty lze vidět na obrázku (Obrázek 28).

4.4.8. Media

Komponenty Media jsou obalující prvky pro obrázky, HTML5 videa a externí videa (ze služeb http://youtube.com aj.). Zaručují, že se tento obsah zobrazí v určitém rozlišení. Defaultně knihovna obsahuje takové komponenty 4 – pro rozlišení 21:9, 16:9, 4:3 a 1:1. Komponenta využívá CSS vlastnost `object-fit`, díky které se obsah zobrazuje nedeformovaně i v případě, že je do komponenty dán obsah s jiným rozlišením, než je dané rozlišení Media komponenty.

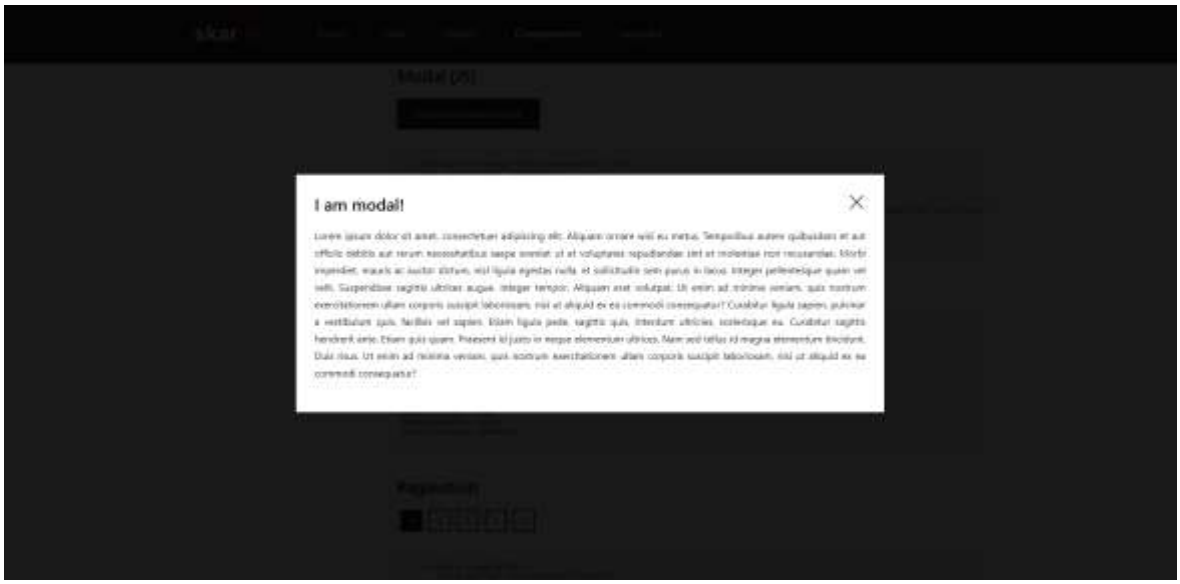


Obrázek 28 - Ukázka komponenty Karta a komponent Media s různorodým obsahem [zdroj autor]

Prohlížeče Internet Explorer a Android Browser však nepodporují potřebnou vlastnost `object-fit` vůbec a prohlížeč Edge ji nepodporuje u HTML5 video elementů. Správné fungování komponenty s obrázkovým obsahem na prohlížeči Internet Explorer knihovna řeší Javascriptem. Pro správné fungování komponenty s video obsahem v prohlížečích Internet Explorer, Edge a Android Browser či s obrázkovým obsahem v prohlížeči Android Browser je nutné dodržet správné rozlišení obsahu – tj. rozlišení obsahu by mělo odpovídat předpokládanému rozlišení obsahu komponenty.

4.4.9. Modální okno

Komponenta modálního okna zahrnuje nejen samotné okno, ale i pozadí za otevřeným modálním oknem a javascriptovou funkcionalitu otevírání / zavírání okna. Pomocí proměnných lze měnit barvu pozadí za modálním oknem, rozměry modálního okna, pozadí okna a vlastnosti křížku – ovládacího prvku modálního okna, po kterém se okno zavře. Okno se zavírá i klikem mimo něj. Při zapnutí modálního okna nelze na stránce scrollovat – to lze pouze v modálním okně.



Obrázek 29 - Ukázka komponenty modálního okna vytvořené knihovny [zdroj autor]

4.4.10. Stránkování

Komponenta stránkování vytváří hotové odkazy na jednotlivé stránky obsahu. Jedná se pouze o nastylované odkazy, samotnou funkcionalitu stránkování je možné řešit například komponentou záložek. Pomocí proměnných je možné měnit barvu odkazů stránkování a rozměry.

Obrázek 30 - Ukázka komponenty stránkování vytvořené knihovny [zdroj autor]

4.4.11. Progress bar

Progress bar je ukazatel postupu. Komponenta samotná je de facto základně nastylovaný HTML progress element. Pomocí proměnných je možné měnit její výšku a barvy. Podpora progress elementu se v různých prohlížečích liší, a tak komponenta nevypadá všude stejně, pracuje však se stejnými barvami i rozměry.

Obrázek 31 - Ukázka komponenty Progress bar vytvořené knihovny [zdroj autor]

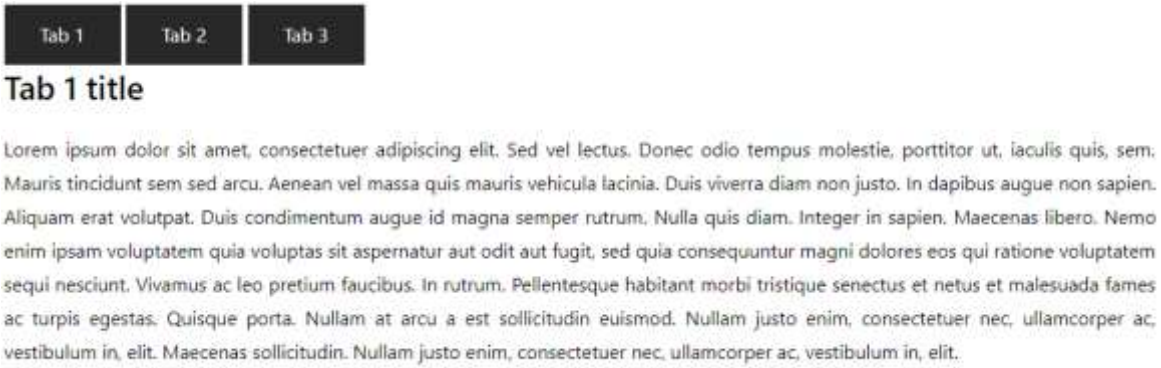
4.4.12. Záložky

Komponenta záložek umožňuje v rámci elementu zobrazování a skrývání jednotlivých obsažených prvků (prvky mají třídu `tab`). Toto zobrazování a skrývání je ovládáno pomocí javascriptu. V případě, že uživatel klikne na prvek ovládající komponentu záložek a zobrazující jeden z obsažených elementů, ostatní elementy se skryjí. Samotná komponenta neobsahuje téměř žádné styly, ale jedná se spíše o javascriptovou funkci. Zápis třech prvků ovládající jednotlivé elementy (záložky) a zápis celé komponenty v jazyce HTML je zobrazen níže:

```
<button class="c-button" data-tabs="tabs-1" data-tab="1">Tab 1</button>
<button class="c-button" data-tabs="tabs-1" data-tab="2">Tab 2</button>

<div class="c-tabs" data-tabs="tabs-1">
  <div class="tab active">
    Obsah záložky
  </div>
  <div class="tab">
    Obsah záložky
  </div>
</div>
```

Kód 15 - Ukázka HTML zápisu komponenty záložek vytvořené knihovny a jejích ovládacích prvků [zdroj autor]



Obrázek 32 - Ukázka komponenty záložek vytvořené knihovny [zdroj autor]

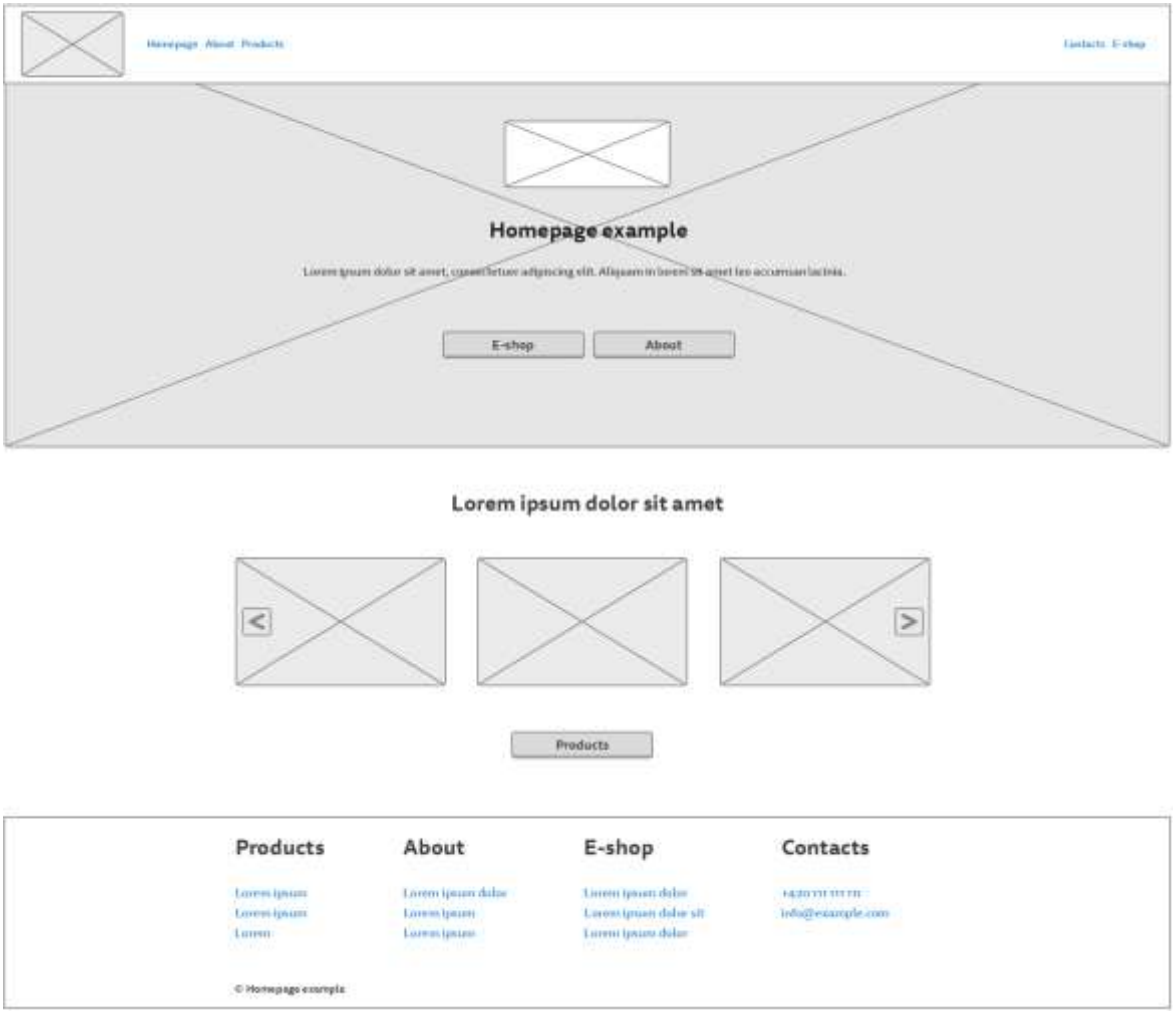
4.4.13. Vizuály

Vizuály jsou komponenty pro obsah s obrázkovým či video pozadím přes celou šířku stránky. Knihovna nabízí jak vizuál s obrázkovým pozadím, tak i vizuál s video pozadím. Pomocí proměnných lze měnit výšku vizuálu a barvu pozadí vizuálu v případě, že nemá nastaveno žádné obrázkové či video pozadí. Jelikož komponenta vizuálu bez obsahu vypadá sama o sobě pouze jako široký obrázek, neuvádí v této kapitole autor obrázkový příklad této komponenty.

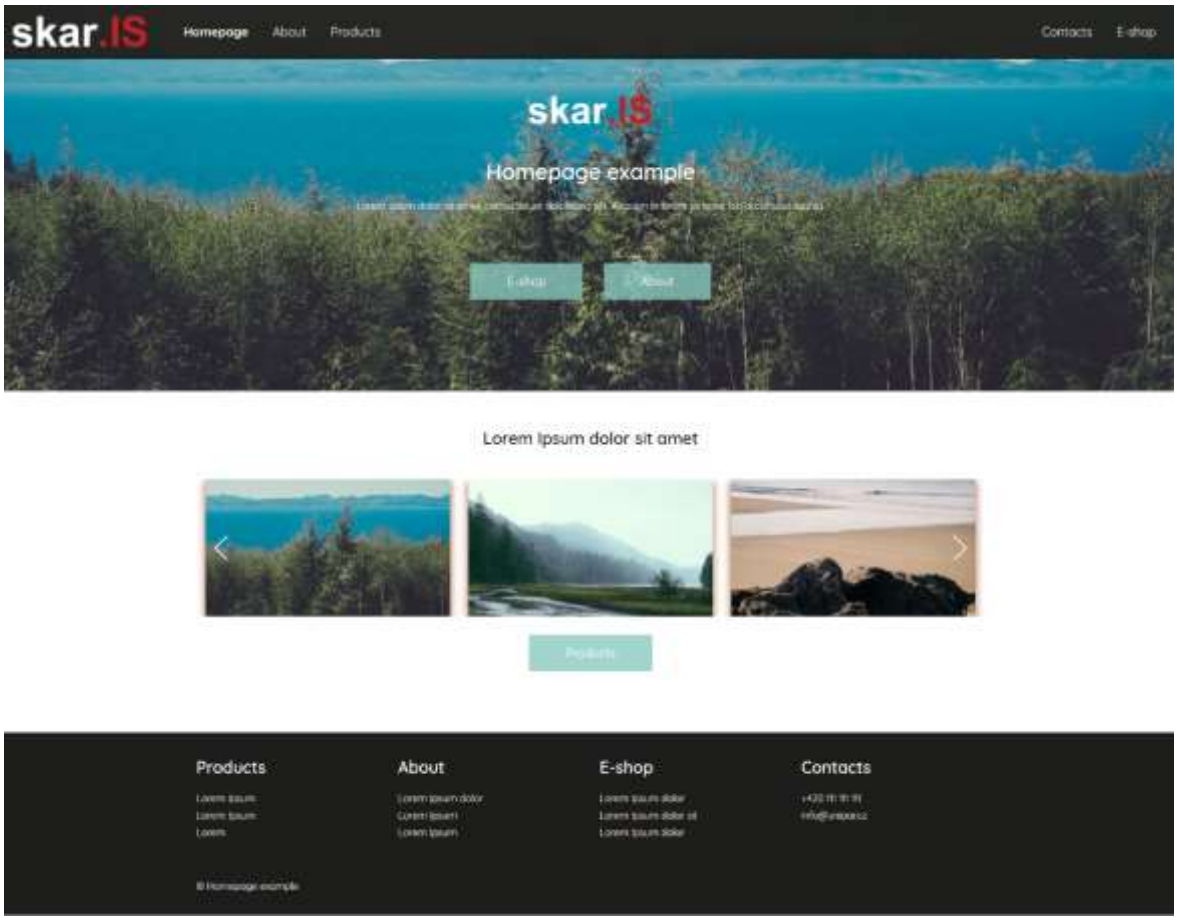
5. Vytvoření příkladů k demonstraci možností vytvořené knihovny

I přesto, že autor v rámci vývoje knihovny ke každé její části vytvořil jednoduché příklady (výsledek některých lze vidět na obrázcích v předchozí kapitole), vytvořil i několik komplexnějších příkladů, které mají simulovat případné další stránky, v rámci jejichž vývoje se knihovna využije. Celkově se jedná o tři příklady, vývoji každého z nich předcházelo vytvoření wireframu.

Prvním příkladem je vzorová úvodní stránka webu, kde jsou použity pomocné třídy, grid systém a komponenty menu, vizuál, tlačítka, *carousel* a patička. Níže lze vidět wireframe příkladu i konečný výsledek.

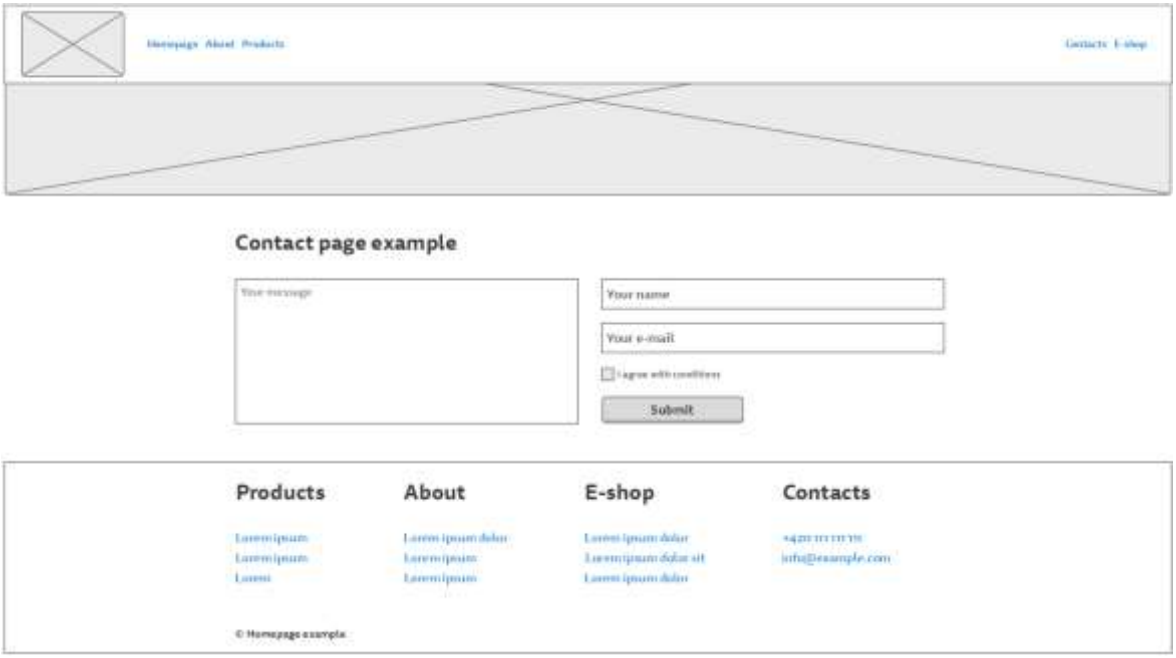


Obrázek 33 - Wireframe příkladu úvodní stránky [zdroj autor]

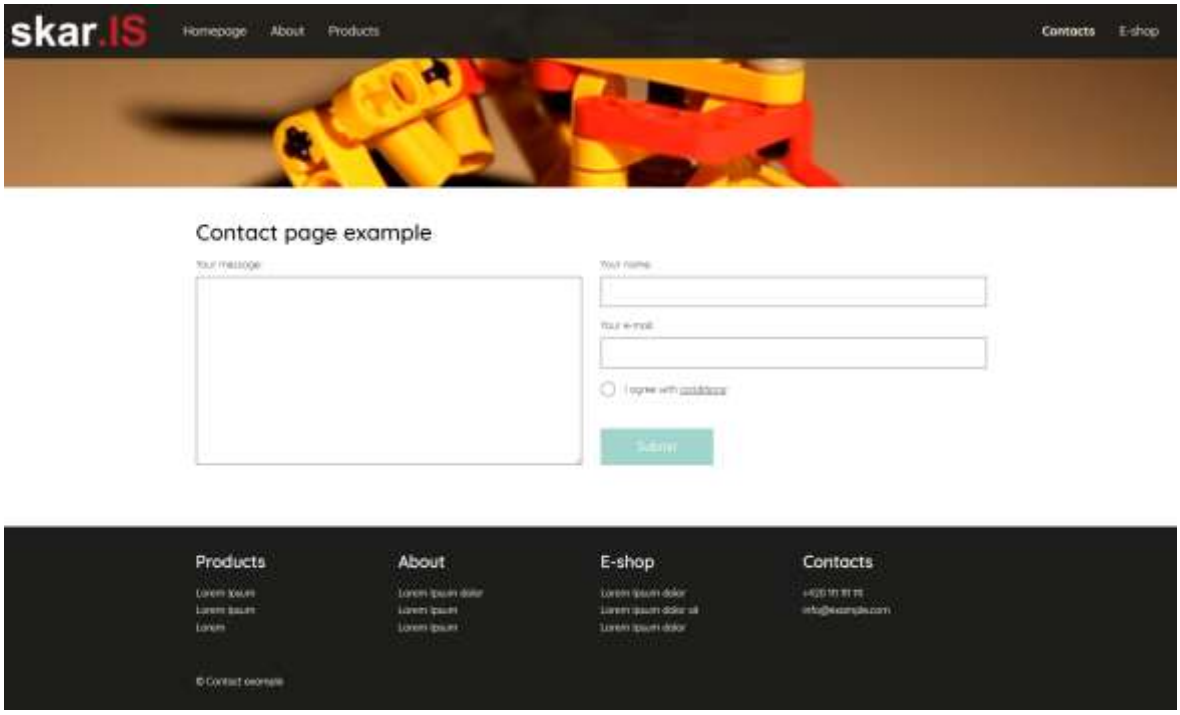


Obrázek 34 - Ukázka vytvořeného příkladu úvodní stránky [zdroj autor]

Druhý příklad vychází z příkladu úvodní stránky – neliší se obsahem menu ani patičkou. Místo ostatního obsahu však obsahuje komponentu video vizuálu a vzorový formulář, ve kterém jsou použity komponenty formulářových prvků a modálního okna. Obsahem příkladu je i vzorová validace, díky čemuž lze vidět, jak případně používat i komponentu chybové hlášky formuláře. Samotná validace však součástí knihovny není. Příklad je nazýván jako *Contact page*, čili ukázka kontaktní stránky.



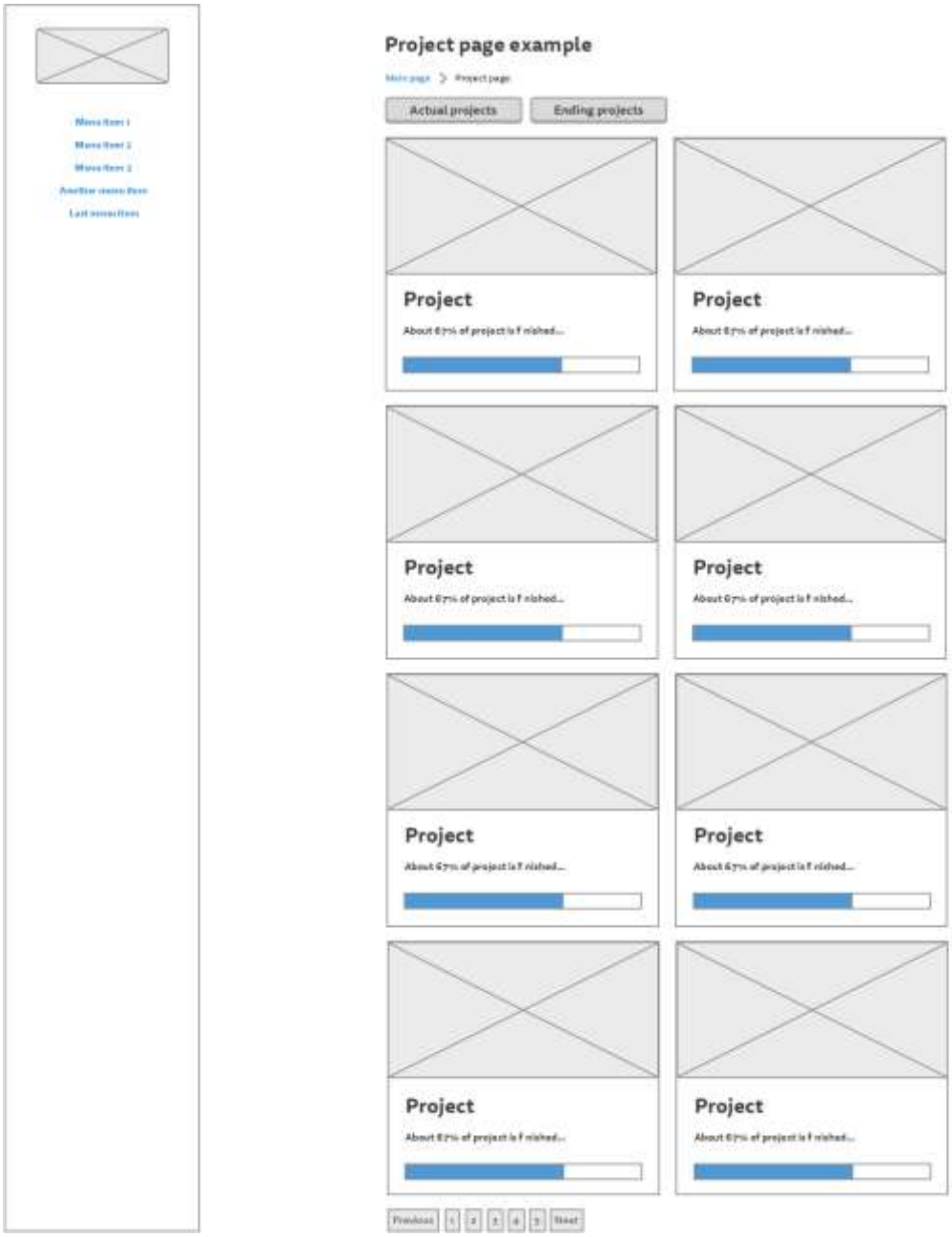
Obrázek 35 - Wireframe příkladu kontaktní stránky [zdroj autor]



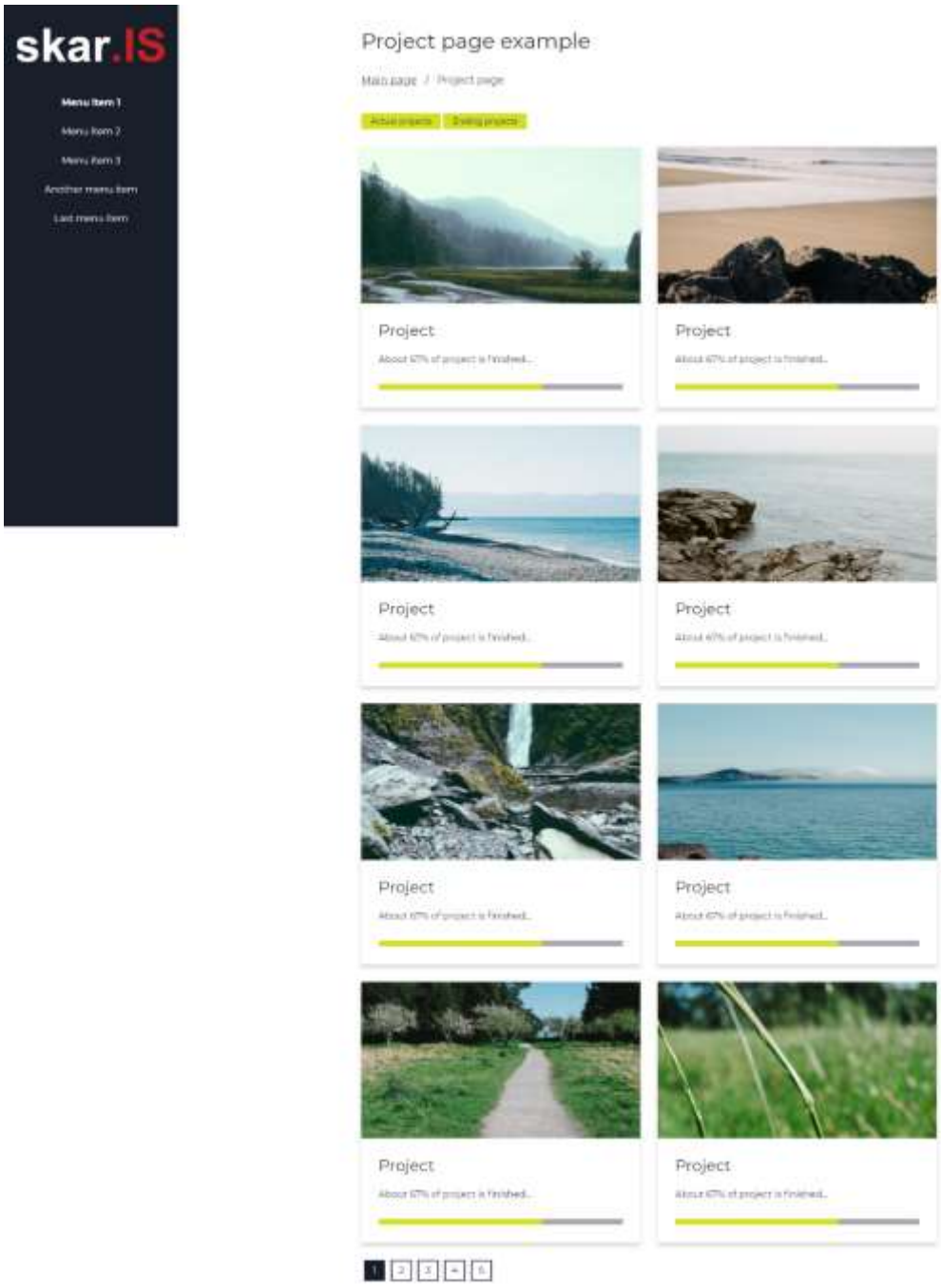
Obrázek 36 - Ukázka vytvořeného příkladu kontaktní stránky [zdroj autor]

Posledním příkladem je pak příklad stránky se seznamem projektů (dále *projektová stránka*). Na ní lze vidět (Obrázek 38), že kontejner obsahu je zde posunutý vpravo kvůli menu umístěnému fixně vlevo. Lze tak vidět, že ani s nevšedním rozložením stránky knihovna nemá problém. Mimo to lze v příkladu vidět použité komponenty vertikálního menu, drobečkové navigace, záložek, karty, medií,

progress baru a stránkování. Na následujících obrázcích nelze vidět, že menu je v reálném příkladu fixní.



Obrázek 37 - Wireframe příkladu projektové stránky [zdroj autor]



Obrázek 38 - Ukázka vytvořeného příkladu projektové stránky [zdroj autor]

6. Vytvoření dokumentace a publikování knihovny

Po vytvoření funkční verze knihovny autor vytvořil komplexní příklady, na kterých bylo otestováno, že pomocí knihovny je možné bez problému vytvořit webovou stránku a že jednotlivé části knihovny fungují i spolu dohromady. Dalším krokem je vytvoření dokumentace pro knihovnu a veřejné publikování celé knihovny.

6.1. Vytvoření dokumentace

Dokumentace pro knihovnu byla vytvořena jako samostatná webová stránka umístěna na adrese http://skaris.skaramart.in. Byla vytvořena pomocí PHP frameworku Symfony a její frontendová část stojí na samotné knihovně skar-is. Dokumentace je rozdělena do pěti sekcí – základy knihovny, grid systém, pomocné třídy, komponenty a příklady. Většina věcí je vysvětlena pomocí příkladů, pro úplné pochopení knihovny se předpokládá, že si vývojář knihovnu stáhne a prozkoumá ji. Dokumentace tak zejména tvoří přehled možností knihovny, dle kterého se může případný vývojář i rozhodnout, zda knihovnu dále prozkoumá. Část s příklady momentálně obsahuje odkazy na jednotlivé komplexní příklady z kapitoly 5. Dokumentace je, na rozdíl od práce, psána v anglickém jazyce[7].

[7] To umožní případné snazší rozšíření mezi vývojáře, jelikož angličtina je primárním jazykem i na serverech GitHub a NPM

Obrázek 39 - Ukázka dokumentace knihovny

6.2. Publikování knihovny

Po vytvoření knihovny, příkladů a dokumentace je knihovna připravena k publikování a k použití ve vývoji. Knihovna je dostupná, jak již bylo psáno v kapitole 3, ze tří zdrojů. Knihovnu lze jednak stáhnout z repozitáře knihovny na serveru GitHub, na který se lze dostat buď rovnou či odkazem ze stránek dokumentace a jednak pomocí balíčkovacích systémů jakými jsou například NPM a Yarn. Z nich je možné knihovnu nainstalovat níže uvedenými příkazy:

```
yarn add skar-is
npm install skar-is
```

Kód 16 - Příkazy balíčkovacích systémů pro instalaci vytvořené knihovny

Knihovna byla publikována pod licencí MIT i v průběhu psaní práce, aktuální verze v době dokončení práce (duben 2018) je 1.1.1.

Závěr

Cílem práce bylo vyvinout vlastní knihovnu, která usnadní tvorbu grafického rozhraní dle různorodého grafického návrhu. Tento cíl byl dále rozdělen na pět dílčích cílů – vytvoření grid systému, znovupoužitelných komponent, pomocných tříd, dokumentace a komplexních příkladů.

Jako první autor analyzoval hotová řešení, z nichž s některými neměl předchozí zkušenosti, a dále se těmito řešeními nechal při vývoji knihovny inspirovat. I díky této provedené analýze následně mohl navrhnout strukturu, vlastnosti i obsah vyvinuté knihovny.

Návrh struktury knihovny byl zcela dodržen. Pomocné třídy i grid systém byly vyvinuty bez větších problémů. Jediným problémem se pak může zdát zajištění fungování některých tříd a grid systému v prohlížeči Internet Explorer, jehož podporu autor považoval za důležitou. Stejně tak byly vyvinuty všechny navrhované komponenty (viz kapitola 4.4). V rámci testování, které proběhlo po vyvinutí celého obsahu knihovny, bylo zjištěno několik problémů, zejména s CSS vlastností `object-fit` a různorodou podporou stylování `<progress>` elementů a formulářových prvků. Komponenty využívající tuto vlastnost či elementy tak nezaručují naprosto stejné chování mezi prohlížeči. Řešením by bylo tuto vlastnost či elementy nepoužívat, nicméně by to stálo větší množství času a autor sám předpokládá, že podpora této vlastnosti či elementů se bude časem zlepšovat a sjednocovat. Vyvinutím všech tří částí knihovny byly splněny první tři dílčí cíle hlavního cíle knihovny.

Stejně tak autor pokládá za splněný i čtvrtý dílčí cíl, tj. napsání dokumentace pro knihovnu. Původně se při vývoji knihovny rozhodoval, zdali napíše dokumentaci přímo do repozitáře (ve formátu MarkDown) či zdali jí věnuje celou webovou stránku. Přistoupil k druhé možnosti, jelikož chtěl, aby součástí dokumentace byly přímo i příklady jednotlivých částí knihovny.

Ve vytvořených komplexních příkladech lze najít použité všechny komponenty knihovny, tudíž autor považuje tyto příklady za poměrně dobrou demonstraci jejích možností. V příkladech sice nejsou moc použity pomocné třídy grid systému a obecné pomocné třídy, ale ty autor nepovažuje pro ukázku tak důležité, jako hotové komponenty. Vytvořením těchto příkladů byl splněn poslední, pátý dílčí cíl. Splněním všech dílčích cílů byl tak naplněn hlavní cíl práce.

Autor by rád v budoucnu knihovnu rozšířil o další komponenty a případně opravil chyby, pokud na nějaké narazí. Dalším plánem bude i postupné zbavování se jQuery závislosti, jelikož vývoj ve firmě Appio směřuje jiným směrem. Autor jQuery využil zejména pro rychlé napsání několika funkcí

komponent. Podle dalších projektů, které bude vyvíjet, uvidí, zdali některé z funkcí nepřepíše do čistého JavaScriptu či jiných dalších javascriptových frameworků.

Knihovna v době dopsání práce (duben 2018) byla použita při vývoji jednoho projektu ve firmě Appio a v jednom autorově vlastním projektu. Dokončení a spuštění těchto projektů autor předpokládá až během května 2018, a tak je v práci bohužel zatím nemohl uvést. Autor doufá, že se mu knihovnu podaří dostat do podvědomí alespoň několika desítek vývojářů. Publikované verze knihovny pod licencí MIT již od 10. 2. 2018, kdy vyšla první rozpracovaná verze knihovny, do 10. 4. 2018 stáhlo několik stovek uživatelů, což může být zapříčiněno tím, že na autorův profil na serverech GitHub a NPM zřejmě uživatelé zavítají kvůli jeho jiným open-source balíčkům. I tak se ale dle autora dá říci, že knihovna vzbudila alespoň malý zájem, i přesto, že pořádné ohlasy získá asi až během následujících měsíců.

Terminologický slovník

Termín	Význam (zdroj)
Autoprefixer	Nástroj pro automatické přidání prefixů pro potřebné vlastnosti CSS. Přidávání prefixů probíhá proto, aby problematické vlastnosti CSS fungovaly i na požadovaných prohlížečích, které vývojář nastaví v nastavení autoprefixeru.
Breakpoint	Bod rozlišení, ve kterém se změní chování či styly určitých prvků na stránce.
Carousel	Komponenta používaná na webových stránkách, ve které se mění obsah ať již automaticky či pomocí ovládacích prvků. (Obrázek 6 - Ukázka komponenty Carousel vytvořené pomocí knihovny Bootstrap. Carousel je všeobecně uznávaný název pro tento druh komponenty. [11])
CSS	Cascading Style Sheets – jazyk ovlivňující vlastnosti zobrazení prvků na webových stránkách. Jeho hlavním smyslem je oddělení vzhledových vlastností od obsahu jednotlivých prvků. [2]
CSS Grid	Modul CSS, sloužící pro jednoduché vytváření komplexních layoutů celé stránky. [5]
CSS knihovna	CSS knihovna je soubor tříd, pravidel a vlastností CSS které mají za úkol pomoct vývojáři při vývoji grafického rozhraní, určitých komponent a jiných řešení. Často pro své fungování využívá i JavaScript. Větší CSS

	knihovny bývají též často nazývány CSS frameworky.
CSS preprocesor	CSS preprocesory jsou jazyky postavené nad CSS, které je následně nutné zkompilovat právě do CSS souborů. Využívány jsou proto, že na rozdíl od jazyka CSS nabízejí více funkcí a zjednodušují díky tomu práci. [7]
Drobečková navigace	Anglicky nazývána breadcrumbs, je navigace znázorňující hierarchii webu. Obsahuje cestu přes nadřazené sekce k sekci momentálně zobrazené.
Fixní pozicování	Pokud je prvek pozicován fixně, znamená to, že zůstává vždy na stejném místě v okně prohlížeče.
Flexbox	Vlastnosti CSS, které určují rodičovskému prvku (Flex container, flex kontejner), jak velké místo v něm budou zaujímat a jak se budou zarovnávat prvky v něm obsažené (Flex items, flex položky). Je tak možné jednoduše měnit zarovnávání prvků do řádku či sloupce, určovat, jak velký bude určitý Flex item oproti ostatním, jak se budou moci měnit flexibilně jeho rozměry, v jakém pořadí prvky řadit apod. [7]
Float	Vlastnost CSS pro obtékání, často se používá i pro zarovnávání prvků vedle sebe.
Font-family	CSS vlastnost definující jaký font se použije pro vykreslení písma.
Grid systém	Struktura, podle které jsou pozicovány jednotlivé prvky a lze díky tomu tak snadno řadit prvky do řádků či sloupců. Pokud se jedná o responzivní systém, manipuluje s umístěním

	těchto prvků dle okna uživatelova prohlížeče (či zařízení). V případě, že se prvky řadí vedle sebe, mluvíme o horizontálním grid systému, pokud pod sebe, mluvíme o vertikálním grid systému. [1]
Hover efekt	Efekt po najetí myší na určitý prvek. Často prvek změní například barvu či jiné vlastnosti.
HTML	HyperText Markup Language – značkovací jazyk využívající se pro tvorbu webových stránek, udává strukturu stránky. Dále s ním lze manipulovat pomocí jazyku JavaScript či udávat prvkům vzhled pomocí jazyka CSS. [2]
JavaScript	Jednoduchý skriptovací jazyk, jehož kód se spouští ve webovém prohlížeči a umožňuje s webovou stránkou dále manipulovat a reagovat na akce uživatele. [2]
jQuery	Javascriptová knihovna, která usnadňuje práci s HTML dokumentem a zjednodušuje zápis některých javascriptových funkcí. [13]
Layout stránky	Rozložení stránky do několika částí s různým významem (např. hlavička, menu, obsahová část, patička, postranní panel apod.).
Margin	CSS vlastnost určující šířku vnějšího okraje prvku.
Media Queries	CSS vlastnosti Media Queries udávají v rámci jazyka podmínky, které umožní aplikovat určité CSS vlastnosti v určitých situacích. Nejčastěji se používají pro definování stylů pro určité rozlišení displeje. [6]

Minifikace souborů	Minifikace souborů spočívá v jejich zmenšování, například se odstraní přebytečné znaky souborů, komentáře a zkracují se zápisy některých funkcí a vlastností.
Mobile-first	Mobile-first je metoda tvorby (či návrhu) webových stránek, kde se primárně stránka vytvoří pro mobilní zařízení a následně se upraví pro uživatele zařízení s větší obrazovkou a desktopů. Přesně opačný přístup je pak nazýván *desktop-first*.
Modální okno	Komponenta používaná na webových stránkách, jedná se o okno, které se uživateli otevře v popředí stránky. Často se zbytek stránky překryje průhlednou barvou. (Obrázek 7)
Open-source software	Open-source software je software jehož zdrojový kód mlže kdokoliv používat, upravovat či vylepšovat. [10]
Padding	CSS vlastnost určující vnitřní okraj prvku.
Pixel, px	Pixel přestavuje jeden svítící bod na monitoru či jeden bod v obrázku a tisku. Pixel nemá pevné rozměry, odvozuje se podle rozlišení.
Rem	Jednotka, která se odvozuje od velikosti písma prvku <html>. Velikost písma v tomto prvku se rovná 1 `rem`.
Responzivita	Responzivní web je takový, který se správně zobrazuje a přizpůsobuje zařízením, které uživatel používá.
SASS	Syntactically awesome style sheets, jeden z nejznámějších a nejpoužívanějších CSS

	preprocesorů. Jako SASS může být nazývána i jedna z jeho syntaxí. [9]
SCSS	Jedna ze syntaxí preprocesoru SASS, velice podobná zápisu jazyka CSS. [9]
Systémové fonty	Nativní fonty operačních systémů, které jsou v nich automaticky předinstalované. Vývojář se tak může při jejich použití spolehnout, že se uživateli na daném operačním systému načtou.
Wireframe	Zjednodušené skica webu, struktura, která popisuje funkcionalitu a rozmístění prvků na stránce. Na rozdíl od grafického návrhu nepracuje většinou s barvami, obrázky a dalšími vlastnostmi prvků, zahrnuje spíše rozložení obsahu a jeho propojení. [22]
Yarn	Yarn je balíčkovací systém Node.js, který stejně jako známější a starší NPM čerpá balíčky a moduly z rozhraní dostupného na adrese www.npmjs.org. Balíčky se instalují pomocí příkazů v příkazovém řádků a jejich struktura je také zapsána v souboru `package.json`. [15]

Použitá literatura

[1] AMRAN, Adam. Úvod do grid systémů. In: Meebio [online]. Meebio, 01.11.2011 [cit. 26-01-2018]. Dostupné z: http://blog.meebio.cz/clanek/158/uvod-do-grid-systemu/

[2] SANDEEP, Panda, Tiffany B. BROWN a Kerry BUTTERS. *HTML5 Okamžitě*. Computer Press, 2015. ISBN 978-80-251-4355-1.

[3] УСАЧЕВ, Максим a fantasai. Evolution of CSS Layout: 1990s to the Future. In: *Fantasai: home* [online]. 2012 [cit. 31-01-2018]. Dostupné z: http://fantasai.inkedblade.net/weblog/2012/css-layout-evolution/

[4] BŘÍZA, Petr. Tvorba layoutu webu – teoretický úvod. In: *Interval.cz* [online]. 2004 [cit. 01-02-2018]. Dostupné z: https://www.interval.cz/clanky/tvorba-layoutu-webu-teoreticky-uvod/

[5] ANDREW, Rachel. *CSS3 Layout Modules* [online]. 2nd edition. United Kingdom: edgeofmyseat.com, 2014 [cit. 07-02-2018].

[6] MICHÁLEK, Martin. *Vzhůru do (responzivního) webdesignu*. Verze 1.1. Praha: vlastním nákladem autora, 2017. ISBN 978-80-88253-00-6.

[7] Vzhůru do CSS3. Martin Michálek, 2015. ISBN 978-80-260-8440-2.

[8] *Can I use... Support tables for HTML5, CSS3, etc* [online]. [cit. 25-02-2018]. Dostupné z: https://caniuse.com

[9] *Sass: Syntactically Awesome Style Sheets* [online]. Hampton Catlin, Natalie Weizenbaum, Chris Eppstein, and numerous contributors, 2017 [cit. 07-02-2018]. Dostupné z: https://sass-lang.com/

[10] What is open source? In: *Opensource.com*[online]. Red Hat [cit. 27-01-2018]. Dostupné z: https://opensource.com/resources/what-open-source

[11] *W3Schools Online Web Tutorials* [online]. W3Schools, 2018 [cit. 20-02-2018]. Dostupné z: https://www.w3schools.com/

[12] Bootstrap · The most popular HTML, CSS, and JS library in the world. [online]. Mark Otto, Jacob Thornton, and Bootstrap contributors [cit. 26-01-2018]. Dostupné z: https://getbootstrap.com

[13] *JQuery* [online]. The jQuery Foundation, 2018 [cit. 08-02-2018]. Dostupné z: http://jquery.com/

[14] ŠKÁRA, Martin. *Výukový kurz jQuery*. Liberec, 2014. Dlouhodobá ročníková práce. Střední průmyslová škola strojní a elektrotechnická a Vyšší odborná škola Liberec. Vedoucí práce Mgr. Michal Stehlík.

[15] *Yarn* [online]. [cit. 09-02-2018]. Dostupné z: https://yarnpkg.com

[16] NIKHIL, John. Facebook's Yarn vs npm—Is Yarn really better?. In: *Medium* [online]. 2016 [cit. 08-04-2018]. Dostupné z: https://medium.com/@nikjohn/facebooks-yarn-vs-npm-is-yarn-really-better-1890b3ea6515

[17] *Gulp.js* [online]. Gulp.js [cit. 08-04-2018]. Dostupné z: https://gulpjs.com/

[18] WICHARY, Marcin. Using UI System Fonts In Web Design: A Quick Practical Guide. In: *Smashing magazine* [online]. Smashing magazine, 13 November 2015 [cit. 18-02-2018]. Dostupné z: https://www.smashingmagazine.com/2015/11/using-system-ui-fonts-practical-guide/

[19] Foundation [online]. Campbell, California: ZURB [cit. 26-01-2018]. Dostupné z: https://foundation.zurb.com/

[20] *Pure* [online]. 2018 [cit. 17-02-2018]. Dostupné z: https://purecss.io/

[21] *Bulma: a modern CSS framework based on Flexbox* [online]. 2018 [cit. 17-02-2018]. Dostupné z: https://bulma.io/

[22] KRATOCHVÍLOVÁ, Viola. JAK SE DĚLÁ WEB? PODÍVEJTE SE, JAK VZNIKÁ WIREFRAME! In: *Aira* [online]. Aira, 2015 [cit. 07-04-2018]. Dostupné z: https://blog.aira.cz/jak-se-dela-web-podivejte-se-jak-vznika-wireframe

Seznam obrázků

Seznam kódů

Příloha A: Elektronické přílohy

Jelikož celá knihovna, příklady i dokumentace mají vlastní repozitář na serveru GitHub, považuje autor za dostatečné na jednotlivé repozitáře odkázat:

- https://github.com/skaryys/skar-is - vytvořená knihovna
- https://github.com/skaryys/skar-is-website-example - vytvořené komplexní příklady, které jsou online dostupné také v rámci dokumentace na adrese http://skaris.skaramart.in/examples
- https://github.com/skaryys/skar-is-documentation - dokumentace knihovny vytvářená jako PHP Symfony aplikace. Jinak dostupná online také na adrese http://skaris.skaramart.in.