Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informatiky a kvantitativních metod

Využití frontend frameworků v javascriptových frameworcích a jejich implementace

Diplomová práce

Autor: Bc. Marie Vinopal Boštíková
Studijní obor: Aplikovaná informatika

Vedoucí práce: prof. RNDr. PhDr. Antonín Slabý, CSc.

Hradec Králové duben 2018

Prohlášení:

Prohlašuji, že jsem diplomovou práci zpracovala samostatně a s použitím uvedené literatury.

……

V Hradci Králové dne 21.4.2018 Marie Vinopal Boštíková

Poděkování:
Upřímně děkuji vedoucímu této závěrečné práce prof. RNDr. PhDr. Antonínu Slabému, CSc. za odborné vedení, cenné rady a věcné připomínky, které mi velmi pomohly při zpracování této práce. Dále bych ráda poděkovala celé své rodině za jejich trpělivost během celého studia a především při dokončování této práce.

Anotace

Cílem této práce je vytvořit souhrnný materiál, který se zabývá oblastí tvorby HTML šablon pomocí CSS frontend frameworku a jejich následnou implementací do javascriptového frameworku.

Teoretická část se zabývá popisem základních rysů CSS a javascript frameworků. Cílem je vysvětlit důvody jejich využívání při budování frontendu aplikací a také zajistit přehled nejznámějších technologií, které se pro tuto činnost používají. Dále se práce zabývá tím, jakým způsobem posupuje kodér při tvorbě HTML šablon, jaké jsou výstupy jeho práce a jaké technologie k tomu používá.

Praktická část se věnuje tvorbě HTML šablony na základě vytvořené grafiky za pomocí dvou konkrétních CSS frontend frameworků. Následně se vytvoří SPA aplikace za pomocí dvou různých javascriptových frameworků. Cílem je implementovat vytvořené HTML šablony do těchto javascriptových aplikací. Práce sleduje náročnost převedení CSS, HTML, JS, a následně hodnotí, zda je nutné práci kodéra změnit, popř. jaké technologie by se měli ideálně používat.

Annotation

Title: Use frontend frameworks in JavaScript frameworks for their implementation

The aim of this thesis is to create a summary material dealing with the creation of HTML templates using the CSS frontend framework and their subsequent implementation into the JavaScript framework.

The theoretical part deals with the description of the basic features of CSS and JavaScript frameworks. The aim is to explain the reasons for their use in creating frontend applications and also to provide an overview of the best-known technologies used for this work. Further, the thesis deals with how the developer provides HTML templates, what are the outputs of his work and what technology he uses.

The practical part deals with creation of HTML template based on prepared graphics using two specific CSS frontend frameworks. Subsequently, SPA applications are created using two different JavaScript frameworks. The goal is to implement created HTML templates into these JavaScript applications. The thesis measures the difficulty of converting CSS, HTML, JS, and which technologies should be ideally used.

Obsah

1 Úvod

1.1 Cíle práce

Cílem této diplomové práce je prozkoumat možnosti implementace HTML šablon nastylovaných kodérem do jednoduché aplikace postavené pomocí javascriptového frameworku nebo knihovny. Dílčím cílem je navrhnout doporučení a principy tvorby HTML šablon pro javascriptové frameworky/knihovny. Práce se též zabývá problémem nezbytnosti naučit se základy javascriptového frameworku, v němž budou použity dodané HTML šablony. V neposlední řadě má práce za cíl vytvořit ucelený přehled o průniku práce kodéra a javascriptového programátora daného JS frameworku tak, aby oba získali přehled o tom, co ke své práci potřebuje jejich protějšek.

1.2 Metodika práce

První, teoretická část práce se zaměřuje na základní rešerši současných trendů v oblasti vývoje HTML šablon pomocí CSS frontend frameworků a také se zabývá základními principy JS frontend frameworků. Čtenář získá přehled o současných technologiích, které se používají při tvorbě frontendové části aplikace.
Praktická část práce se již zaměřuje na konkrétní využití zmíněných technologií zmíněných v teoretické části. Popisuje implementaci jednoduché grafiky do jednotlivých javascriptových frameworků.

1.3 Rešerše

Současné odborné knihy či bakalářské a diplomové práce se tématem implementace HTML šablon postavených nad CSS frontend frameworky do javascriptového frameworku v podstatě vůbec nezabývají. Řeší buď zobrazení frontendu jako takového (např. responsivní webdesign a trendy v oblasti vývoje HTML, CSS) nebo se naopak zaměřují jednotlivé JS frameworků a jejich použití – např. vytváření SPA aplikací. Praktická část se proto opírá hlavně o dokumentace jednotlivých technologií.
Dalšími zdroji v diplomové práci jsou elektronické knihy či online články na webových stránkách.

2 CSS Frontend framework

Framework již má jasně danou strukturu, definuje základní postupy vývoje, nabízí připravená řešení a umožňuje soustředit se vývojářům na vývoj klíčových funkcionalit aplikace.

CSS Frontend framework můžeme popsat jako soubor připravených knihoven a částí zdrojového kódu, které mají za cíl usnadnit práci při vývoji HTML šablon pro frontend webových stránek (aplikací).

2.1 Frontend aplikace

Frontend aplikace můžeme jednoduše definovat tím, co daný uživatel vidí na obrazovce počítače/prohlížeče. Vzhledem k tomu, že se tato práce primárně zabývá webovými projekty, jedná se o zobrazení aplikace v internetovém prohlížeči.

Aby dokázaly prohlížeče zobrazit zamýšlený vzhled aplikace, potřebujeme k tomu tyto tři základní technologie/nástroje.

1. HTML – zdrojový kód šablony
2. CSS – kaskádové styly
3. JS – javascript

2.1.1 HTML

Webová stránka je jednoduchý textový soubor napsaný v HTML (HyperText Markup Language). HTML popisuje obsah i strukturu dokumentu pomocí označení různých prvků dokumentu. Například hlavní nadpisy, podkapitoly, odstavce, číselné seznamy, popisy obrázků atd. (1)

2.1.1.1 Historie HTML

V počátcích HTML neexistovala žádná organizace, která by se věnovala určováním pravidel či standardizace syntaxe tohoto jazyka. Vývojáři webových prohlížečů si interpretovali HTML dle svého, což mělo za následek různé zobrazení stejné stránky v několika jiných prohlížečích. Tento problém začala řešit organizace W3C (World Wide Web Consortium), která spojila přední webové designéry i programátory. Tato organizace

začala určovat pravidla a soubory standardů, nicméně bez možnosti vynutit si jejich dodržování. To znamená, že ne vždy byla tato pravidla hned implementována do všech prohlížečů. Ohledně vydávání dalších verzí HTML se vedly a vedou diskuze, které trvají i několik let a není vždy snadné je rozhodnout.

Můžeme uvést příklad toho, že tvorba standardů pro HTML (frontend) není jednoduchá. W3C přišla s novou vizí verze XHTML 2.0. Nicméně tato vize nebyla příliš příznivě přijata a nově vzniklá skupina WHATWG přišla s vlastní myšlenkou na novou verzi HTML označovanou jako HTML5. Cílem skupiny WHATWG, do které se řadí mnoho předních technologických firem (Google, Microsoft, Mozilla) (2), je návrh nových standardů, které předkládá W3C ke schválení. Verze HTML5 byla předložena W3C a po řadě diskuzích nahradila koncept XHTML 2.0.

Jedním z hlavních důvodů, proč HTML5 bylo více podporováno, byla zpětná kompatibilita, kterou chtěla XHTML 2.0. částečně zrušit. V tuto chvíli je verze HTML5 v drtivé většině aktuálních prohlížečů podporována a vývojáři frontendu již HTML5 aktivně používají při vývoji. (3) (4)

Tabulka 1 - Verze HTML

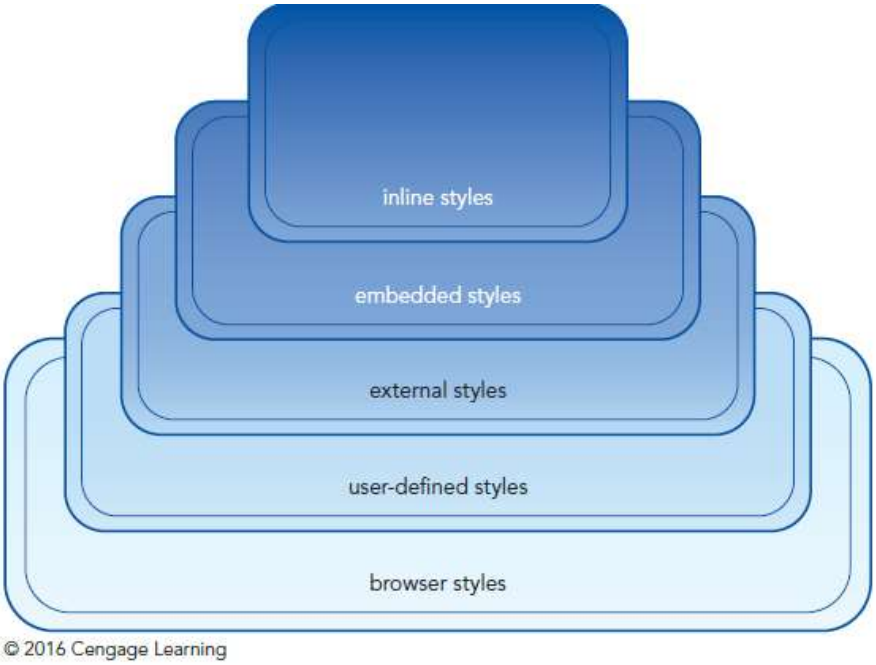
Verze	Rok	Popis
HTML 1.0	1989	První publikovaná verze HTML
HTML 2.0	1995	Přidány interaktivní elementy včetně webových formulářů
HMTL 3.2	1997	Verze poskytla další podporu pro webové tabulky a rozšířila možnosti interaktivních prvků formuláře a skriptovacího jazyka
HTML 4.01	1999	Verze, která podporovala vytváření souborů stylů a poskytla webdesignerům větší kontrolu nad rozložením a vzhledem stránky a umožnila podporu multimediálních prvků, jako jsou audio a video
XHTML 1.0	2001	Verze přeformulovala předchozí verzi pomocí značkovacího jazyka XML s cílem poskytnout vynutitelné standardy pro HTML obsah a umožnit interakci HTML s jinými XML jazyky
XHTML 2.0	skončila v 2009	Verze navržená kvůli opravě některých chyb a problémů spojených s převzetím syntaxe HTML 4.01
HTML 5.0	2012	Aktuální verze HTML poskytující podporu pro mobilní design, sémantické prvky, sloupcový design, validaci formulářů, offline ukládání a rozšířená multimédia

Zdroj: převzato z (1)

2.1.2 Cascade Style Sheets (CSS)

Cascade Style Sheets – kaskádové styly formátují HTML a upravují jeho vzhled v prohlížeči. Stylování se provádí pomocí definovaných atributů (stylů) a jejich vlastností. Tyto styly mohou být implementovány buď:

1. Inline – přímo v rámci HTML elementu
2. Pomocí <style> tagu v hlavičce HTML dokumentu
3. Externě – pomocí .css souboru, který se nalinkuje do HTML stránky
4. Styly v prohlížeči
    a. Základní styly prohlížeče – user agent stylesheet (např. nadpis má H1 nastavenou velikost 2em.)
    b. Uživatelské styly v prohlížeči (např. defaultní velikost písma)



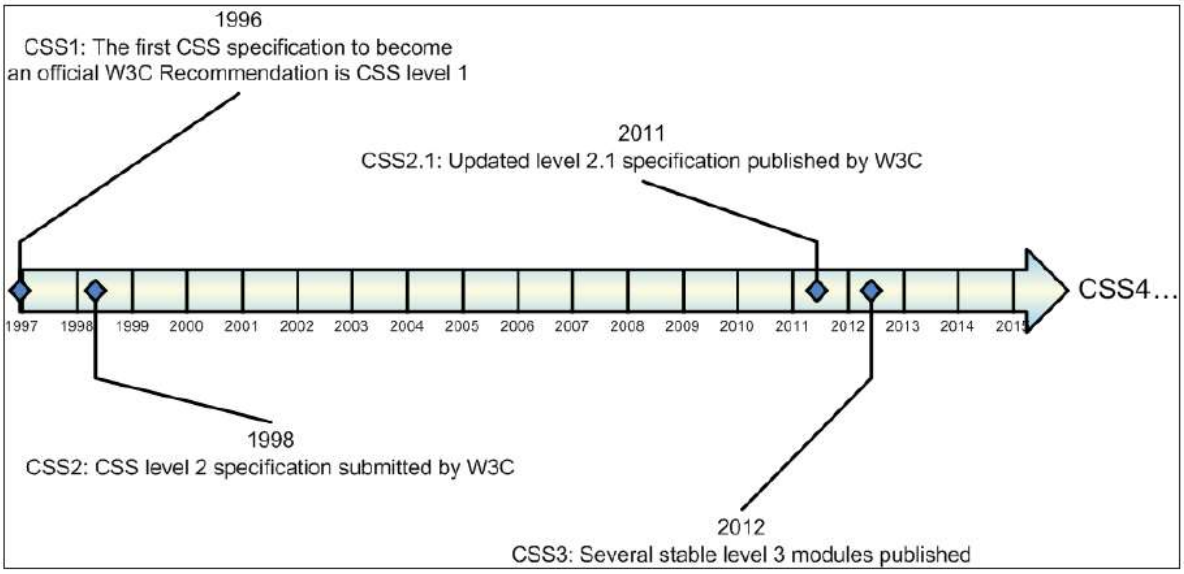
Obrázek 1 - Typy stylů – zdroj (1)

2.1.2.1 Historie CSS

Pravidla a specifikace pro CSS jsou spravovány podobně jako HTML organizací World Wide Web (W3C). Koncept CSS byl poprvé navržen v roce 1994. Cílem CSS bylo poskytnou jednoduchý deklarativní jazyk, který bude flexibilní jak pro autory, tak i pro uživatele. „Kaskády" (dědění definovaných vlastností) umožňují kombinace stylů pro jednotlivé prvky s tím, že je mohl ovládat jak vývojář, tak i uživatel v prohlížeči (např. základní zvětšení písma). (5)

Již v roce 1996 byla vydána verze CSS1 a začalo se pracovat hned na další verzi CSS2. Nicméně se zde narazilo na obdobný problém jako u HTML, kdy vývojáři webových prohlížečů nedokázali dostatečně rychle implementovat nové standardy pro CSS, a navíc je některé prohlížeče interpretovaly různě. V tomto směru dochází k velkým rozdílům v zobrazení webové stránky napříč prohlížeči. Developeři jsou nuceni se učit specifika jednotlivých prohlížečů (tzn. jakým způsobem daný CSS styl interpretují). Tento problém přetrvává řadu let. Ani vydání verze CSS2, která byla vydána v roce 1998, tuto situaci nijak nezlepšilo. Samotná verze byla ještě upravena na verzi CSS2.1 a doporučena k finální implementaci do prohlížečů až v roce 2011 – což dokazuje, že byl opravdu problém

se shodnout na definicích ohledně CSS. Po dokončení první verze CSS2 se začalo pracovat na novém konceptu CSS3, který si vzal za cíl modularizovat CSS, aby se některé důležité moduly mohly vyvíjet rychleji než ostatní a rovnou nasazovat. Při využití předchozí filozofie nasazování verzí by museli developeři čekat na dokončení všech ostatní modulů, což by mohlo trvat neúměrně dlouho. (6)



Obrázek 2 - Historie vývoje CSS – zdroj: (7)

2.1.2.2 CSS – současnost

Aktuální verze CSS je CSS3, která je zpětně kompatibilní s CSS2.1. CSS3 obsahuje řadu modulů, které ještě nejsou podporovány v řadě prohlížečů, nicméně jejich základní moduly jsou již stabilní a lze tedy říci, že CSS3 lze v této době bez problémů používat při vývoji frontendu. Samozřejmě vývojáři musí počítat s tím, že například podpora některých CSS3 modulů ve starších verzích prohlížečů není v podstatě žádná a je riskantní ji používat (např. flexbox).

Vývojáři v této oblasti musí být ostražití a vycházet z analýzy faktu, zda se jejich aplikace bude spouštět spíše na nových zařízeních s novými/aktuálními prohlížeči, nebo se bude pouštět i na několik let starých počítačích. Nicméně tyto problémy by měly být v blízké budoucnosti méně významné nebo dokonce odeznít. Microsoft vydal upozornění uživatelům, kteří používají starší verze prohlížeče Internet Explorer, že již je nebude nadále podporovat. (8)

Jiná situace může být i v některých velkých firmách, kde mohou na počítačích mít z řady historických důvodů nainstalovány staré vrze prohlížečů, které jsou kompatibilní s některými neaktualizovanými aplikacemi.

2.1.3 JavaScript (JS)

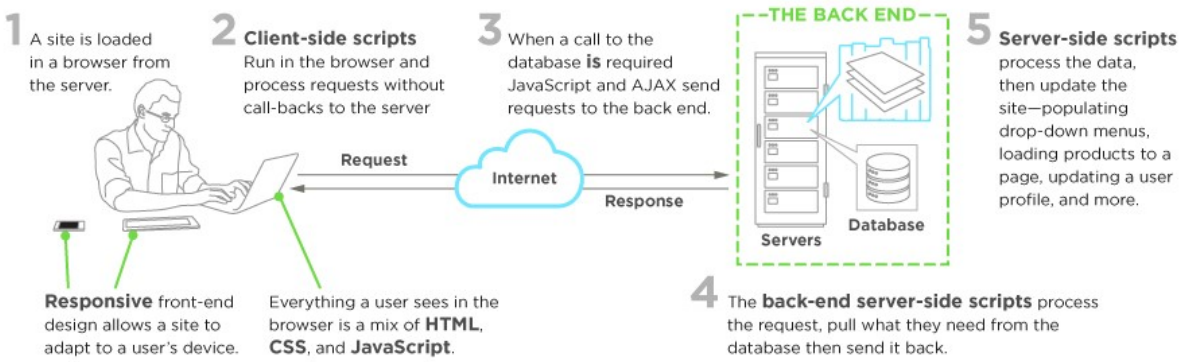
Javascript je programovací jazyk, který se řadí mezi objektově orientované jazyky. Jeho primární použití je na straně klienta (webový prohlížeč), nicméně se javascript používá v této době i jako serverový jazyk (node.js).

2.2 Cesta ke standardizaci vývoje frontendu

Díky rychlému rozvoji výše uvedených technologií a raketovému nárůstu používání internetu vznikl velký tlak na vývojáře i grafiky, aby zajistili uživatelům webových aplikací komfort, na který byli doposud zvyklí z desktopových (nativních) aplikací.

Webové stránky a aplikace vyvíjí velké množství firem, které se samozřejmě snaží maximalizovat zisk, proto usilují o to definovat standard práce na webových aplikacích, kterými jsou právě frameworky.

Práce na novém projektu pro frontend developera probíhá zpravidla stále stejně. Podílí se na tvorbě grafiky a návrhu frontend funkcionalit (UX). Musí převést grafický návrh do HTML šablon (HTML, CSS, JS). Následně šablony předává vývojářům backendu a pomáhá jim při „oživování“ šablon – napojování na backend aplikace. Následně pomáhá udržovat a rozvíjet frontend aplikace, nicméně již není jeho účast úplně nutná a jeho kapacity může zaplnit vývoj jiného projektu. (9)



Obrázek 3 - Front End Development – zdroj: (9)

Překotný vývoj frontendu, který probíhá i v současné době nenabízí velký prostor pro vznik stabilních frameworků, které bude rozvíjet dostatečně silná komunita. Zde zaznamenáváme velký rozdíl oproti backendovým frameworkům, které na tom jsou z pohledu udržitelnosti a rozvoje daného frameworku na mnohem lepší úrovni (PHP – Nette, Symfony; Java – Spring, …, .NET).

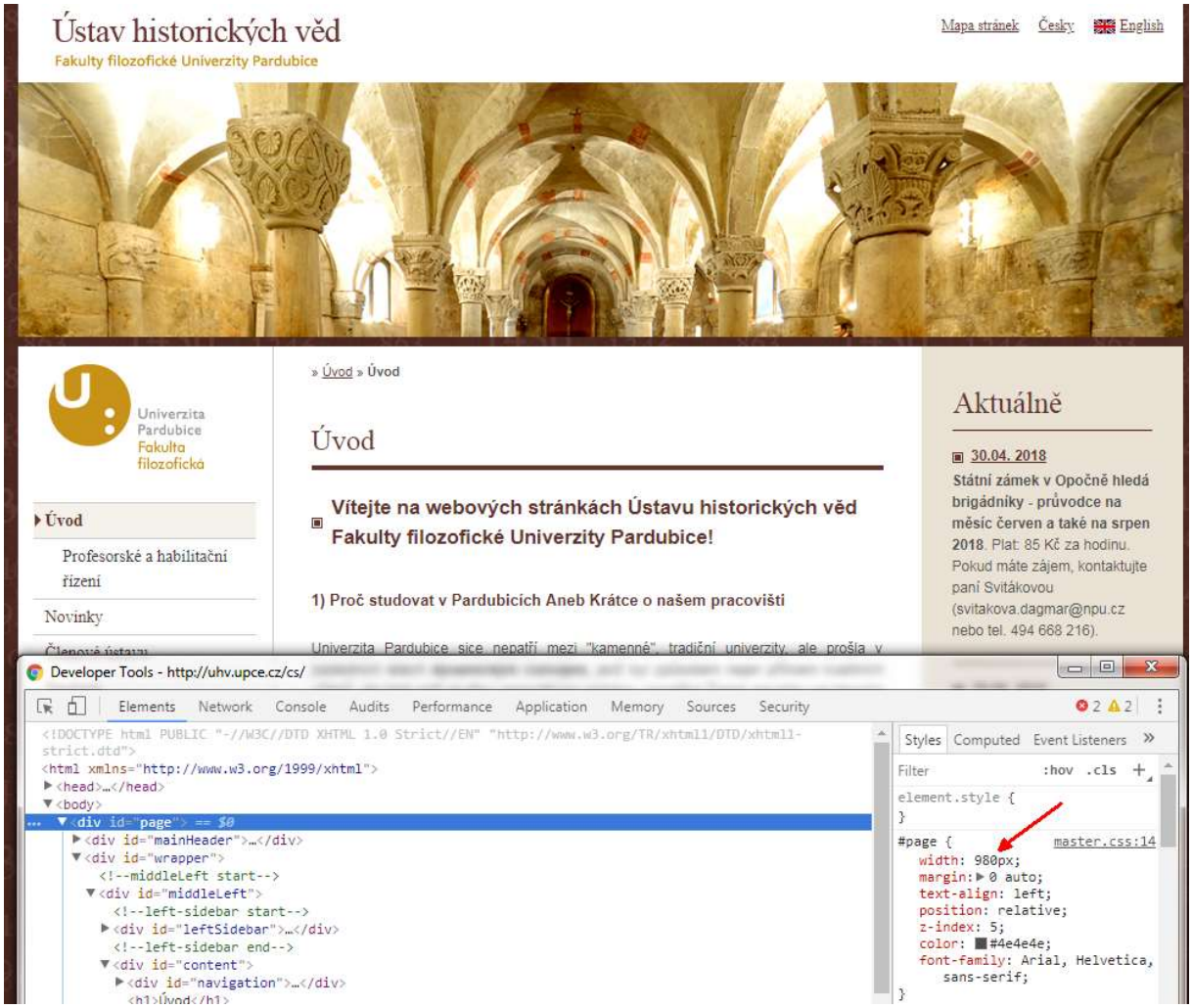
Frontend vývojáři museli a stále musí reflektovat rozvoj technologií na straně backendu a frontendu i interpretaci jejich kódu napříč různými operačními systémy, prohlížeči i zařízeními (mobil, tablet, počítač, hodinky) a musí zjištěné změny implementovat do jejich development workflow.

I přes výše uvedené okolnosti stále platí, že standardizace vývoje frontendu se ve většině projektů rozhodně vyplatí a použití, popř. vývoj jakéhokoli frontend frameworku usnadní jak jeho vývoj, tak i jeho následné oživení backend vývojáři a jeho následný support.

2.2.1 Responsive design

Termín „responsivní webový design“ vytvořil Ethan Marcotte v roce 2010. Sloučil tři do té doby existující techniky (flexible grid layout, flexible images/media a media queries) do jedné, kterou nazval responsivní web design. (10)

Dříve měl web standardně nadefinovanou pevnou šířku, která byla většinou okolo 960px. Pevná šířka se využívala především proto, aby stránka udržela svůj vzhled napříč různými rozlišeními monitorů a aby grafici mohli navrhnout grafiku, která byla nadefinována právě na pevné rozměry. (11)



Obrázek 4 - Příklad pevné šířky webové stránky – zdroj: http://uhv.upce.cz/cs/ - upraveno autorem

V současné době jsou ve velké míře starší frontendy přepisovány do moderního responsivního designu. Jedním z hybatelů tohoto vývoje je společnost Google, která se snaží zlepšit tímto způsobem komfort svých uživatelů. Společnost Google při vyhledávání na mobilním telefonu začala upřednostňovat weby, které jsou responsivní a jejich obsah se bude v mobilním zařízení správně zobrazovat.

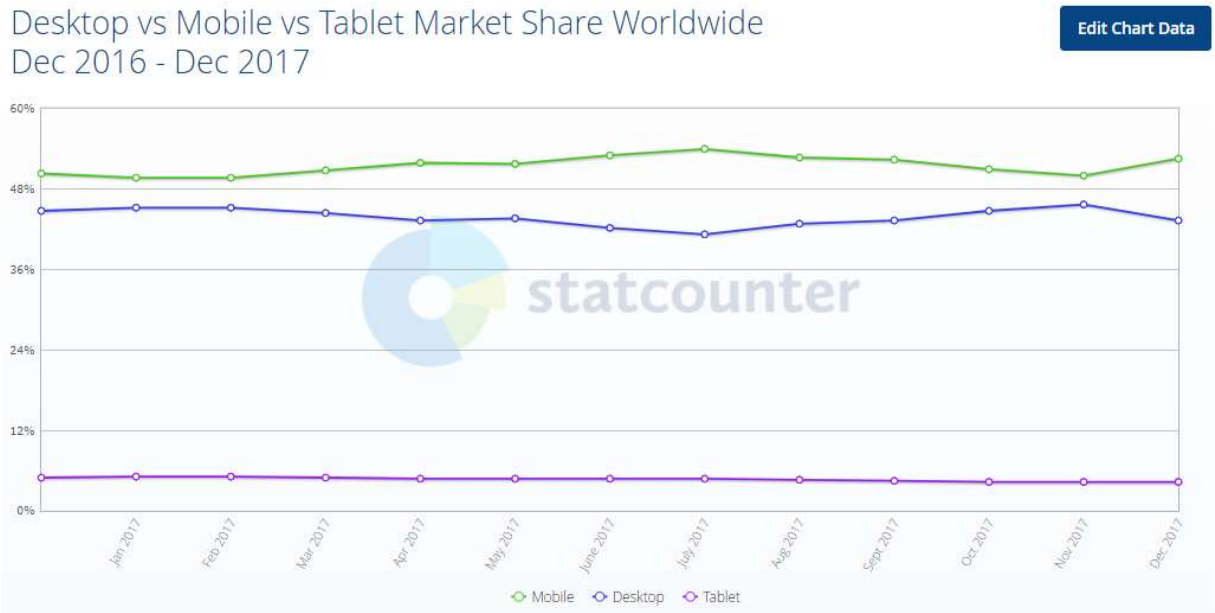
V roce 2007 představila společnost Apple svůj revoluční iPhone a ukázala nové možnosti prohlížení webových stránek a lze říci, že tím změnila přístup a interakci s webovými stránkami.

2.2.1.1 Použití mobilních prohlížečů

Poměr použití mobilních telefonů při prohlížení webového obsahu neustále roste. Mezi rokem 2010–2011 došlo k nárůstu používání globálního mobilního prohlížeče ze 2,86 % na 7,02 %. V roce 2015 se zvýšilo na 33,47 %.

Obrázek 5 - Vývoj prodejů mobilních a dalších zařízení v roce 2015 - zdroj: (12)

Na konci roku 2017 se používal dokonce mobilní prohlížeč více (52,48 %) než klasický desktopový (43,26 %).

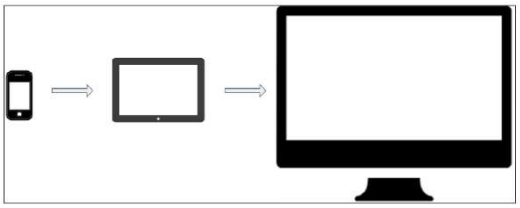


Obrázek 6 - Vývoj prodejů mobilních a dalších zařízení v roce 2017 - zdroj: (12)

Na trend používání mobilních telefonů při prohlížení webových stránek zareagovali vývojáři i výrobci. Mobilní zařízení a jeho displej se zvětšují a prohlížeč zpřístupnil funkcionalitu, která umožňuje vývojářům frontendu pracovat s šířkou obsahu v závislosti na šířce/rozlišení displeje zařízení. Navíc tato zásadní funkce je implementována jen v prohlížeči zařízení, není k tomu potřeba žádná backendová služba na straně serveru, která by tuto funkci zajišťovala. (13)

2.2.1.2 Mobile-first philosophy

Mobile-first filozofie popisuje přístup k přípravě grafiky a tvorbě webové šablony (HTML, CSS) i obsahu, a to od mobilního rozlišení po standardní desktopové rozlišení (klasický monitor). Opačný přístup (degradace) má velký dopad na přepisování stylů i javascriptů konkrétně pro dané mobilní či tablet zařízení.



Obrázek 7 - Mobile First Principe – zdroj: (7)

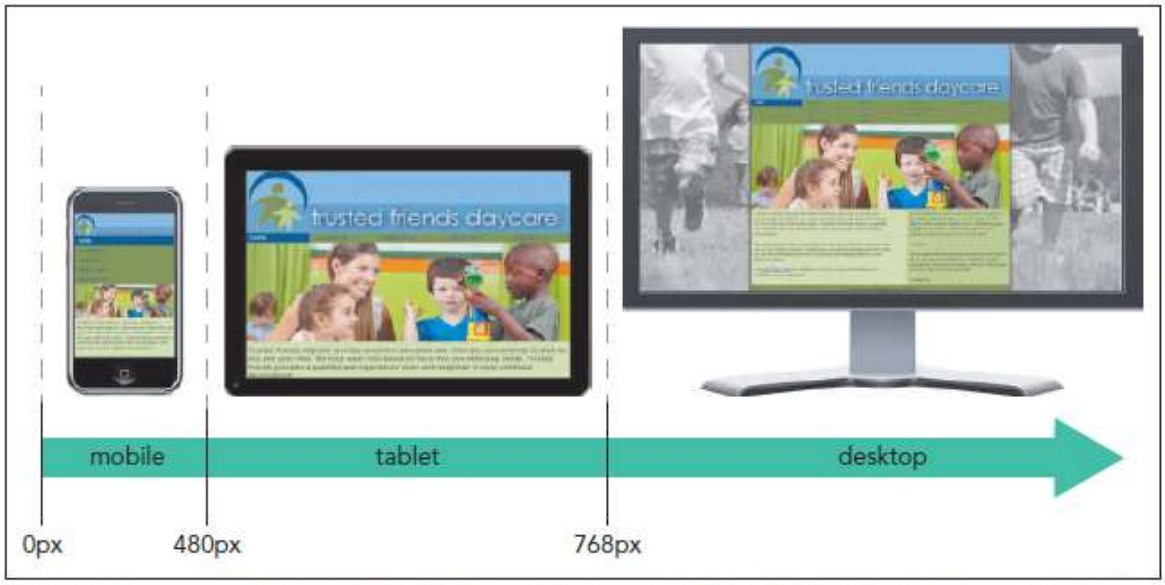
Mobile-first princip je revoluční přístup, ale není dosud standardem ve vývojářských firmách. Problém začíná už u grafiků, kdy je klientem v první řadě požadován návrh desktopové vizualizace, a končí vývojáři, kteří stále spíše pracují spíše s degradací při psaní webových šablon.

CSS3 umožňuje vývojářům snadno využít jeho vlastností pro implementaci mobile-first přístupu do vývoje webových šablon (frontendu) a v kombinaci s některými z CSS front-end frameworků může vývojář vyvíjet šablony dle této filozofie bez velkých zásahů do jeho dosavadního stylu práce.

Právě responsive design pomohl vzniku CSS frontend frameworků, které pomáhají developerům odřídit základní chování šablon v jednotlivých rozlišeních displejů od mobilních zařízení po klasické monitory.

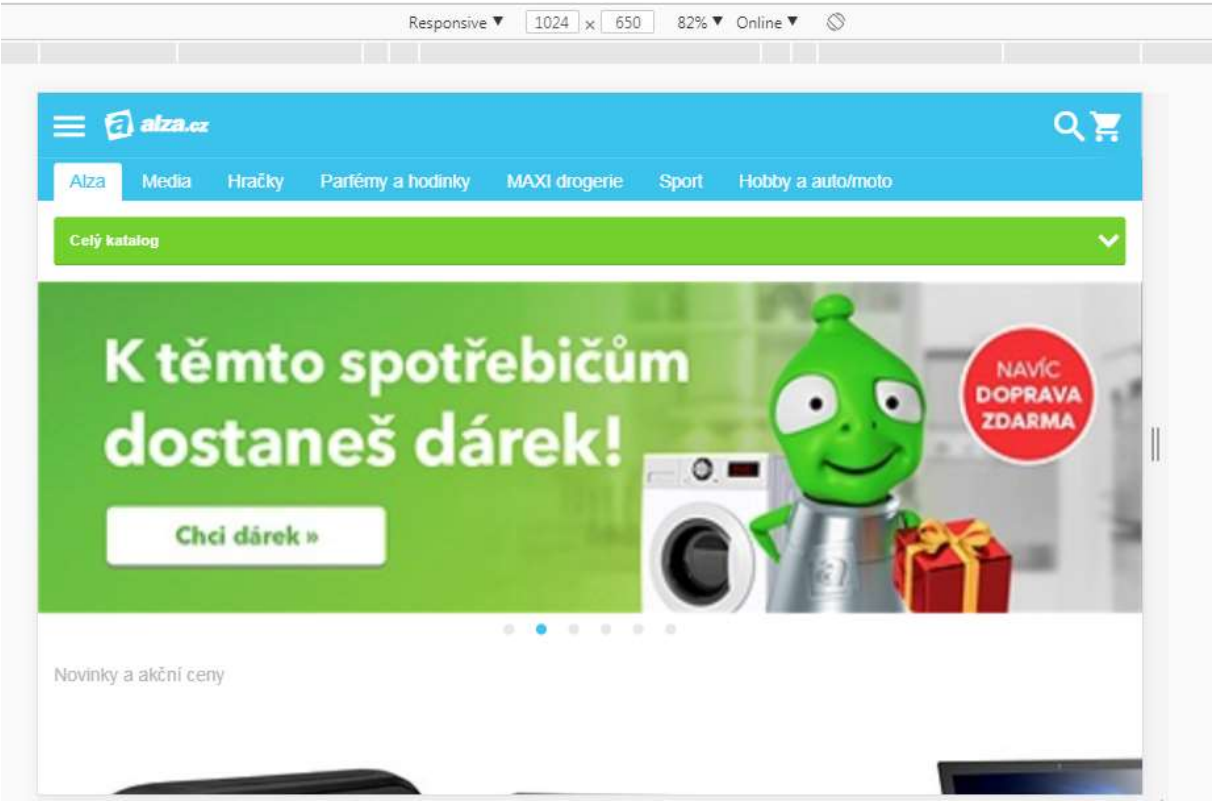
2.2.1.3 Media Queries – breakpointy

Media queries je vlastnost CSS3, která umožňuje cílit specifické CSS styly v závislosti na možnostech daného zařízení (výška, šířka displeje – otáčení zařízení: landscape vs. portrait). Například pomocí několika řádků CSS můžeme změnit vzhled pozadí stránky pro mobilní či desktopové zařízení. (13)



Obrázek 8 - Media Queries Principe – zdroj: (1)

Díky této vlastnosti mohli začít vývojáři uvažovat o tom, že by se existující mobilní webové aplikace projektu mohly zrušit (typicky: m.nazevwebu.cz). S použitím responsivního webdesignu již šlo totiž napsat frontend aplikace, takovým způsobem, aby se zobrazoval správně na všech zařízeních (mobil, tablet, desktop) a není třeba udržovat dvě aplikace, které zobrazují ta samá data. Nicméně některé projekty jsou natolik specifické, popř. mají tak odlišný design i funkcionalitu, že převedení do responsivního designu by bylo příliš náročné a těžko udržitelné.



Obrázek 9 - Příklad mobilní verze stránky – rozlišení 1024px – zdroj: https://m.alza.cz/

2.3 Využití CSS frontend frameworku

Jak již bylo zmíněno v předchozí kapitole, využití CSS frontend frameworku umožňuje vývojářům standardizované postupy při vývoje HTML šablon. Následující podkapitoly postihují základní vlastnosti, které obsahuje většina CSS frontend frameworků.

2.3.1 Standardizace zápisu HTML

CSS frontend framework definuje základní komponenty, s kterými se každý vývojář webové aplikace již setkal. Tyto komponenty mají nadefinované HTML, stačí tedy použít předpřipravený zdrojový kód frameworku v právě vyvíjené aplikaci.

Typy komponent, které framework většinou obsahuje:

- Formulářová pole (input, label, textarea, button, …)
- Grid (jedná se o systém, který umožňuje snadnější přípravu HTML šablony podporující responsivní chování v prohlížeči)
- Menu (navigace – horizontální/vertikální)
- Štítky (novinka, akce)

- Informační panely (alerty)
- Modal - (vyskakovací okno)
- Accordion, tabs – překlikávání obsahu

2.3.2 Standardizace zápisu CSS

Stávající vzhled již existujících komponent, popř. layoutu ocení spíše backend vývojáři, kteří se příliš nezabývají vzhledem. Definované styly se dají v dobrém frontend frameworku jednoduše změnit a lze je konfigurovat dle potřeb implementované grafiky. Při vývoji stylů pro nové komponenty se dá inspirovat již existujícími styly a pokračovat v jejich logice (viz. Dokumentace daného frameworku)

2.3.3 Standardizace zápisu Javascript – pluginy, hotová řešení

Javascript je již nedílnou součástí webových aplikací, takže mít pouze HTML a CSS ke komponentám nestačí vždy a uživatel již v některých situacích očekává funkcionalitu zajištěnou javascriptem. Frontend framework se snaží i v této oblasti postihnout nejčastější funkce, aby je nemusel developer psát stále znovu.

Jedná se především o:

- Modal – vyskakovací okno
- Accordion, tabs, toggle, dropdown
- Tooltip
- Carousel – slider

2.3.4 Důvody k využití či nevyužití CSS frontend frameworku

Obecné důvody pro využití frontend frameworku jsou již popsány výše, nicméně ne vždy je nutné CSS frontend framework použít a je tedy vždy potřeba provést analýzu daného projektu a rozhodnout dle předem definovaných kritérií, která jsou uvedena dále v části 2.3.4.1 a 2.3.4.2.

2.3.4.1 Důvody proč nevyužít CSS frontend framework

1. Jedná se o jednoduchou stránku bez složitých komponent a s minimem obsahu (microsite, landing page)
2. Stránka bude mít pouze statický obsah – nebude se napojovat na backend systém
3. Jedná se pouze o komponentu, nikoli o celý web nebo stránku (např. kontaktní formulář)

Pokud alespoň jeden z výše uvedených bodů platí, je možné, že náklady spojené s konfigurací frameworku budou vyšší než samotné napsání HTML šablony pro frontend bez frameworku.

2.3.4.2 Důvody proč využít CSS frontend framework

1. Na vývoji frontendu bude pracovat více frontend developerů
2. Projekt se bude napojovat na backend systém
3. Je provoz aplikace plánován na období delší než rok?
4. Dostane interní tým dodanou aplikaci do správy?
5. Bude mít webová aplikace více jak 4 typové stránky?
6. Bude webová aplikace responsivní (mobil, tablet, desktop, …)?

Pokud platí více než dva uvedené body, měli by vývojáři frontendu definovat frontend framework pro vyvíjenou aplikaci. Buď si napíší vlastní nebo využijí již existující CSS frontend frameworky.

2.4 Nejpoužívanější CSS frontend frameworky

Tato kapitola obsahuje seznam nejznámějších CSS frontend frameworků, které se používají při přípravě frontendu webových stránek a aplikací. U některých je uveden i příklad použití, které vyžaduje základní znalost technologií, jež dnes musí každý frontend developer znát (npm, yarn, node.js, git, …). (14) (15)

2.4.1 Bootstrap

Jedná se jeden z nejznámějších a nejpoužívanějších CSS frontend frameworků.

Dokumentace: https://getbootstrap.com/docs/4.1/getting-started/introduction/

Git: https://github.com/twbs/bootstrap

2.4.1.1 O frameworku

V srpnu 2010 vydala společnost **Twitter** framework Bootstrap jako open source. Bootstrap je od roku 2012 nejvýznamnějším projektem na síti GitHub. V lednu roku 2018 vydal Bootstrap nejnovější verzi, která obsahuje řadu nových přístupů. Asi největším z nich je změna CSS preprocesoru z původního LESS na SASS. Dalším posunem pro bootstrap je použití flexbox vlastností v gridu layoutu. Kromě řady předpřipravených komponent (viz. níže) Bootstrap zajišťuje responsivní grid, který se velmi snadno používá a v případě, že grafika počítá s tímto gridem, se hlavní rozložení stránky (desktop, mobile) tvoří velmi snadno. (7)

Základní charakteristiky frameworku:

1. Postaven na mobile-first přístupu
2. Responsive design
3. Automatická podpora a ovládání prohlížeče
4. Snadné přizpůsobení frameworku

2.4.1.2 Použití frameworku

Začít stavět webovou stránku na Bootstrap frameworku je velmi jednoduché. Lze použít již zkompilovanou verzi a tu vložit do vytvářeného projektu a pouze prolinkovat příslušné soubory (.css a .js). V případě, že se jedná o rozsáhlejší projekt, lze stáhnout zdrojové kódy Bootstrapu a vytvořit si vlastní build. Jakým způsobem se implementuje tento framework bude předvedeno v praktické části.

2.4.1.3 Bootstrap – dostupné komponenty

V případě, že by bylo nutné upravovat proměnné, které Bootstrap používá, nebo by bylo třeba měnit framework tak, aby odpovídal specifickým požadavkům daného projektu, je nutné stáhnout si celý projekt Bootstrapu do vlastního projektu a následně ho zkompilovat. Výhodou je, že je možné začít používat preprocesory CSS a nakonfigurovat potřebné i nepotřebné komponenty pro daný projekt. To má pozitivní dopad například na velikost CSS souboru. (16)

Bootstrap se používá v řadě firem, které vytváří webové stránky. To znamená, že je jeho využití opravdu celosvětové. Např. platforma Wordpress nabízí velmi širokou škálu šablon postavených právě nad Bootstrapem. (17)

2.4.2 Foundation

CSS frontend framework podobně jako Bootstrap vznikl v rámci skupiny vývojářů soukromé firmy Zurb. Tato společnost se snažila standardizovat přístupy k tvorbě frontendu svých projektů. Když v roce 2011 zjistili, že se jim povedlo dát dohromady ucelený frontend framework, rozhodli se ho publikovat a poskytovat ho nadále jako open-source. V roce 2016 rozšířili Foundation o projekt, který nabízí framework na tvorbu responsivních HTML emailů. (18)

Dokumentace: https://foundation.zurb.com/sites/docs/

Git: https://github.com/blai/foundation

2.4.2.1 O frameworku

Foundation, podobně jako ostatní CSS frontend frameworky, nabízí řadu komponent a základní layoutová řešení. Jedná se o rozsáhlý projekt, který vývojářům frontendu pomůže definovat přístup k psaní frontendu aplikace a ušetří čas strávený na vývoji aplikace.

Základní charakteristiky frameworku:

1. Mobile-first přístup
2. Responsive design
3. Framework pro tvorbu responsivních emailů
4. Zajištění aktualizace projektu – je aktivně vyvíjen firmou, která framework používá na stovkách svých projektů

Foundation používá jQuery, které je stále velmi rozšířené a oblíbené mezi frontend developery. To znamená, že framework je velmi oblíbený a lze ho najít v řadě projektů. Je plně kompatibilní s backend technologiemi. Je velmi často využíván ve známých redakčních systémech jako je Wordpress nebo Drupal.

3 JS frontend framework

Rozvoj javascriptových frameworků úzce souvisí s rozvojem samotného javascriptu. Ještě před několika lety vznikaly knihovny, které pomáhaly vývojářům psát jednotný kód pro všechny prohlížeče (např. jQuery). Tyto knihovny se staraly o správnou interpretaci javascriptu v prohlížeči. Byl velký rozdíl psát například nějakou funkcionalitu v javascriptu pro prohlížeč Firefox a pro starší verzi Internet Exploreru. V případě, že se někdo rozhodl vyvíjet aplikaci postavenou primárně na javascriptu, byl nucen neustále sledovat aktualizace jednotlivých prohlížečů a některé části javascriptového kódu psát speciálně pro daný typ prohlížeče, což znepřehledňovalo kód aplikace a ovlivňovalo stabilitu a dostupnost aplikace napříč prohlížeči.

Podobně jako u CSS frontend frameworku, má i JS frontend framework stejný cíl – pomoci vývojářům standardizovat jejich práci a v mnoha směrech ji usnadnit. Javascriptové frameworky jsou z pohledu tradičních backend frameworků vcelku nové a stále nedochází ke snížení počtu těchto frameworků. Naopak stále vznikají frameworky nové, které se snaží přitáhnout pozornost vývojářů po celém světě.

JS framework se snaží standardizovat práci s Javascriptem na úrovni frontendové části aplikace. Framework se stará o návrh architektury aplikace i o datové přenosy a jejich zpracování. Dá se na něj také nahlížet jako na soubor javascriptových knihoven, které usnadňují vývojářům práci. Řeší velmi rychle se rozvíjející oblast SPA (single-page-application) (19), kdy se javascript stará o vykreslení obsahu dané stránky ve webovém prohlížeči, a to na základě dat, která dostane ze serveru. Následně mění obsah jen té části stránek, kde došlo ke změně (např. nově příchozí zprava), aniž by musel danou stránku celou aktualizovat. Vývoj SPA aplikací je podpořen nejen řadou JS frameworků, ale velmi dobrou podporou javascriptu v nejpoužívanějších prohlížečích. Javascript je již několik let na vzestupu a dá se používat nejen pro tvorbu frontendu, ale i k psaní backendových funkcionalit (node.js).

Některé hojně používané javascriptové projekty developery po celém světě se nemusí přesně vejít do striktní definice frameworku a jedná se spíše o knihovny javascriptu. Nicméně část komunity o těchto nejznámějších projektech mluví jako o frameworku, i když to není zcela přesné – jedná se např. o React nebo jQuery. (20)

3.1 Single page application (SPA)

Jak již bylo zmíněno výše, v SPA se vše odehrává v rámci jedné stránky. Změny obsahu na stránce může iniciovat jak uživatel, tak server. Komunikace mezi frontendem a serverem probíhá na pozadí, aniž by docházelo k opětovnému načtení celé stránky. Primárně se pro přenos dat používá AJAX, který zpravidla vrací data ve formátu JSON. Data jsou následně interpretována JS frameworkem a dochází k vykreslení daného obsahu pomocí HTML. Koncept SPA je velmi populární také z toho důvodu, že se snaží řešit dostupnost aplikací i na mobilních zařízeních. (21) Z předchozího textu vyplývá, že podíl uživatelů mobilního webu je již cca 55 %. Tito uživatelé očekávají rychlou a pohodlnou aplikaci, která bude funkční při napojení na pomalejší internet, případně se snadno vypořádá s jeho nestabilitou. Znamená to, že se na mobilní zařízení zobrazí pouze data potřebná právě pro mobilní vzhled.

3.2 Využití JS frontend frameworku

JS frontend frameworky zásadně pomáhají a určují trendy v práci s Javascriptem v prohlížeči. Využitím některého z JS frontend frameworku ušetří vývojáři, kteří se rozhodli pro implementaci SPA, především čas a zajistí si podobně jako u backend frameworků základní standardy při rozvoji dané aplikace.

Rozhodnutí o tom, jaký JS frontend framework využít, je velmi důležité. Nejedná se už o malou javascriptovou knihovnu s několika metodami, ale o robustní nástroj, který určuje směr, jakým bude aplikace vyvíjena. V případě, že se aplikace bude rozšiřovat a rozvíjet delší dobu, může se špatná volba frameworku stát velmi nepříjemnou záležitostí jak pro vlastníka aplikace, tak i pro její vývojářů. Někdy z analýzy vyplyne, že není potřeba mít k dispozici přímo JS framework, protože by se nemuselo vyplatit ho implementovat. Je možné, že bude vývojářům stačit dobrá javascriptová knihovna, která aplikaci posune technologicky kupředu.

Na rozdíl od frameworků jsou javascriptové knihovny méně robustní, většinou řeší konkrétní problematiku a v neposlední řadě jsou mnohem snazší k pochopení a naučení. JS frameworky již mají předem definovanou adresářovou strukturu, mají vlastní koncepce, konvence a pravidla, které je nutné dodržovat. Je zde mnohem větší úroveň abstrakce než u knihoven. Křivka učení JS frameworků je rozhodně delší. Pomocí

frameworku lze rychle vybudovat 80 % projektu nicméně zbylých 20 % může zabrat mnohem více času. (20)

Oblasti a otázky, které by neměly být při hledání a výběru JS frameworku/knihovny přehlédnuty (19)**:**

- Komunita vytvořená kolem konkrétního frameworku
- Četnost aktualizací
- Křivka učení: čím je křivka učení pro nový nástroj vyšší, tím větší jsou náklady na zavedení frameworku
- Podpora ostatních knihoven – je třeba myslet na kompatibilitu používaných knihoven (i vlastních)
- Závislosti/Dependencies
- Modularita
- Jak velký je projekt?
- Jaké vzorové návrhy doposud tým vývojářů používal?
- Jak moc je projekt složitý?
- Kolik je času na dokončení projektu?

3.3 Nejpoužívanější JS frontend frameworky

Při hledání nejpoužívanějších či nejznámějších javascriptových frameworků/knihoven je určitě nejlepší vycházet z aktuálních článků na internetu, které se tomuto tématu věnují. Je jich totiž opravdu hodně a lze téměř konstatovat, že jich snad začíná být více než samotných developerů. Například nejznámější repository síť Github, obsahuje informace asi o čtyřech milionech projektů, které používají javascript. (22) V rámci komunity javascript vývojářů se nejvíce prosadilo těchto několik frameworků a knihoven. (23)

1. React
2. Angular 2
3. Vue.js
4. Ember.js

Následující podkapitoly mají za cíl popsat základní informace o jednotlivých javascriptových frameworcích.

3.3.1 React

React byl vytvořen ve společnosti Facebook a v roce 2013 byl vydán pro volné použití jako open source projekt. React je považován za knihovnu, nikoli za kompletní framework jako např. Angular2. (24) Zpravidla je zařazován v architektuře MVC pod písmeno V jako View, protože React neobsahuje Model ani Controllery. Stará se primárně o uživatelské rozhraní a jeho renderování. (25)

React vnesl do frontend komunity nový způsob uvažování o stavbě a provozování frontendu aplikace. Vize vývojářů z Facebooku byla tedy ze začátku kvůli tomu přijata spíše opatrněji a vyvolávala rozporuplné reakce.

React je založen na myšlence, že manipulace s DOM je příliš náročná a měla by být co nejvíce minimalizována. React to vyřešil pomocí virtuálního DOM, který při zjištění změny dat mění pouze tu část DOM, které se změna týká (26). HTML kód stránky generuje React čistě pomocí javascriptu, i když se může zdát, že jeho syntaxe vypadá téměř stejně jako standardní HTML.

V dnešní době aktivně používá React nejen Facebook, ale i celá řada dalších společností. Ty buď přepsaly existující aplikace do Reactu, nebo postavily nové aplikace již nad Reactem. React se prosadil především u aplikací, kde je velmi vysoká návštěvnost webů spojená s častou změnou dat v průběhu návštěvy uživatele stránek. React se již také ve velké míře prosazuje i při implementaci v oblasti e-commerce řešení. To znamená, že se vývojářům povedlo začít aktivně používat React u standardních byznys projektů.

Příklady projektů používající React:

- https://www.facebook.com/
- https://www.instagram.com/
- https://bitbucket.org/
- https://avocode.com/
- http://www.wix.com/
- https://www.coursera.org/

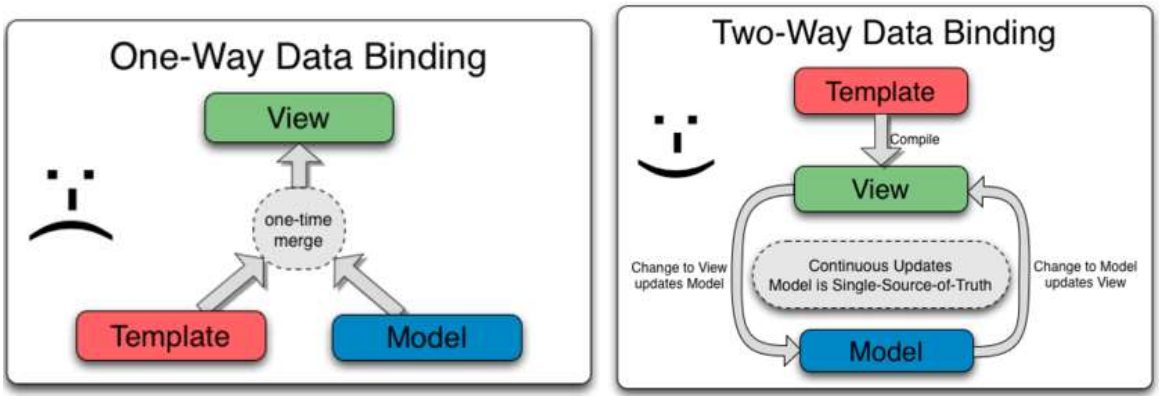
- https://www.wimdu.com/
- https://www.netflix.com

React je z pohledu na stáří ostatních javascriptových knihoven stále relativně nový. Nicméně budoucnost určitě má. Především se bude stále zaměřovat na co nejrychlejší vykreslování a aktualizace stránek. Další oblastí rozvoje Reactu je jeho část, tzv. React Native, která umožňuje vývojářům napsat iOS nebo Android aplikací za pomocí Reactu. To znamená, že dokáže ušetřit čas při vývoji multiplatformních aplikací a zároveň udržet základní koncepci vývoje nad vybranými platformami. (24)

3.3.2 Angular 2

Angular byl vytvořen pracovníky společnosti Google v roce 2009. Angular je, stejně jako React, open source projekt. Angular zapadá do architektury MVVM (model-view-ViewModel), popř. MVC (model-view-controler).

Angular podporuje tzv. obousměrné provázání dat (two-way data binding). (27) To v praxi znamená, že pokud se změní hodnota na uživatelské úrovni, typicky formulářové pole (input, textarea), tak dochází k aktualizaci dat v Modelu. A funguje to i opačným směrem. Takže pokud dojde ke změně dat v modelu, automaticky dojde i ke změně frontendu (view).



Obrázek 10 - Provázání dat – zdroj: (27)

V tuto chvíli je vydaná aktuální verze Angular 2. Ten je nástupcem Angular 1, který se stal jedním z nejrozšířenějších javascriptových frameworků. Nicméně jeho nedostatky začaly omezovat výkon a škálovatelnost aplikací, které na něm byly postaveny. Angular 2 se

snažil všechny problémy Angular 1 vyřešit a nabízí vývojářům moderní nástroj pro tvorbu moderního webového frontendu. Angular 2 podporuje TypeScript, ES6, ES5, CoffeeScript, a dokonce i Dart. Pro vývojáře Angular 1, kteří spravovali na této verzi frameworku své projekty, nebylo úplně příjemné zjištění, že Angular 2 je od základu přepsaný framework. V důsledku toho je čekala velmi náročná reimplementace, pokud chtěli do své současné aplikace implementovat vyšší verzi. (28)

Framework umožňuje psát i nativní aplikace např. pro Android nebo iOS, podobně jako React Native. V případě Angularu se jedná o platformu Nativescript. (29) Tato platforma umožňuje psát nativní aplikace nejen za pomocí Angularu, ale i TypeScriptu.

4 Tvorba HTML šablon pro frontend webové aplikace

HTML šablony zpravidla vytváří kodér (frontend developer). Tato pracovní pozice je relativně mladá a důvody jejího vzniku jsou logické. Programátor programuje backend aplikace (funkcionalita a logika) a tvorba frontendu ho zbytečně zdržuje od jeho hlavní pracovní náplně. Jak bylo zmíněno v kapitole číslo 2, rozvoj technologií v rámci frontendu jde velmi rychle kupředu a backend programátor nemá dostatek času sledovat trendy a ani nemá zpravidla chuť řešit pozice a barvy jednotlivých elementů v HTML. Proto tuto práci začali řešit pracovníci, kteří měli jisté znalosti o HTML, CSS a trochu o Javascriptu. Nejsou to programátoři v pravém slova smyslu, takže zpravidla nemají tak vysoké platové nároky a mohou pracovat bez problémů i externě.

Kodér se tedy primárně zabývá vytvářením HTML šablon na základě dodaného grafického návrhu, případně se spolupodílí na tvorbě funkcionalit frontendu přímo s grafikem.

Díky tomu, že programátor delegoval tvorbu HTML šablon na specializovaného pracovníka, bylo možno změnit i přípravu podkladů pro vytvářený projekt. Grafik a kodér mohou společně vytvořit statickou verzi aplikace, která není napojena na backend aplikace. Takto lze snáze odhalit nedostatky při zobrazování šablon napříč různými zařízeními a také lze dopředu zkontrolovat zamýšlené chování aplikace na frontendu.

V případě, že jsou HTML šablony schváleny, předávají se programátorům k „oživení“ – to znamená, že se HTML šablony navážou na jednotlivé komponenty užitého frameworku, který se stará o renderování frontendu. Z tohoto pohledu lze uvažovat, že buď bude dodané HTML šablony oživovat programátor tradičních MVC frameworků nebo programátor javascriptu v některém JS frameworku.

4.1 Jakým způsobem dodává kodér HTML šablony?

Výše bylo popsáno základní použití CSS frontend frameworků, nicméně příprava HTML šablon vyžaduje v dnešní době více technologií než samotný CSS frontend framework. Pro výrobu HTML šablon (HTML, CSS, JS) používají kodéři řadu technologií, které se starají o zpracování všech potřebných úkonů. Samozřejmě se jedná o pokročilé techniky, které by měl ovládat každý kodér, přestože tomu tak často není. (30)

4.1.1 Balíčkovací systém pro Node.js (npm)

Balíčkovací systém pro node.js se stal standardem pro odřízení závislostí (dependencies) vývojového prostředí HTML šablon. Stará se nejen o dotažení node.js modulů, ale také o stažení balíčků, které se používají přímo na frontendu aplikace. (31)

4.1.1.1 Instalace NPM a základní použití

Na počítač je třeba nainstalovat node.js. Následně se pomocí příkazového řádku spustí inicializace npm v rámci projektu, která vytvoří potřebný package.json soubor. Ten pak slouží, k uložení konfigurace dependencies projektu. (32)

```
> npm init
```

Po spuštění příkazu, lze na základě postupné instalace doplnit základní informace, které se mají v package.json uložit.

```
> npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg> --save` afterwards to install a package and
   save it as a dependency in the package.json file.

   Press ^C at any time to quit.
   name: (test) test-npm-init
   version: (1.0.0)
   description: Description....
   entry point: (index.js)
   test command:
   git repository: not-yet
   keywords: test
   author: Marie Vinopal Boštíková
   license: (ISC)
```

Následně lze přidat jakýkoli npm balíček

```
> npm install <package_name>
```

Pokud je třeba uložit npm balíček do package.json, což je ve většině případů žádoucí, je nutné přidat potřebný parametr.

```
> npm install <package_name> --save
```

Níže je uveden příklad instalace známé javascriptové knihovny jQuery.

```
> npm install jquery -save
```

Ukázka 1 - Výsledný package.json se automaticky rozšířil o dependency na jQuery

```
{
  "name": "test-npm-init",
  "version": "1.0.0",
  "description": "Description....",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "repository": {
    "type": "git",
    "url": "not-yet"
  },
  "keywords": [
    "test"
  ],
  "author": "Marie Vinopal Boštíková",
  "license": "ISC",
  "devDependencies": {
    "jquery": "^3.3.1"
  }
}
```

Velkou výhodou využití NPM je nejen snadná správa verzí jednotlivých balíčků, ale také snížení fyzické velikosti celého projektu. V případě, že se v rámci projektu tvorby HTML šablony nepoužívá NPM, popř. jiný balíčkovací systém (Bower), je nutné ukládat celé soubory knihoven, což je velmi nepraktické a z dlouhodobého hlediska neudržitelné.

4.1.2 Task manažer (Task runner)

Task manažer se stará nejen o přehlednost, ale právě o funkcionality, které dříve kodér požadoval po backend programátorech (např. optimalizace souborů .css). Jedná se především o rutinní operace, jako např. kopírování souborů, build preprocesorů CSS (less, sass, postcss), build javascriptů (browserify, coffescript, concat, minifikace, babel), spouštění navolených akcí, dle potřeby vývoje HTML šablon. (33) V rámci editoru lze nastavit build řady preprocesorů, nicméně následně vzniká problém s nastavením mezi vývojáři, kterých může být více a každý může používat jiné vývojové prostředí (IDE). V tomto směru se jedná o velmi dobrou standardizaci práce kodérů na frontendu. Mezi nejznámější task manažery patří: Grunt, Gulp a Webpack. (34)

4.1.2.1 Příklad implementace Grunt do projektu

Grunt je třeba nainstalovat dle návodu na počítač a vzhledem k tomu, že se jedná o node.js aplikaci, lze jej nainstalovat pomocí NPM.

```
> npm install -g grunt-cli
```

Následně je ještě nutné nainstalovat všechny závislosti.

```
> npm install grunt –save-dev
```

V rootu projektu je potřeba vytvořit soubor Gruntfile.js. Obsah souboru lze definovat pomocí ukázkové konfigurace v dokumentaci Gruntu (35). Aby byla ukázka fungování gruntu kompletní, je třeba nadefinovat vybrané tasky, které se mají provést. Jedná o task, který zařídí build less souboru do CSS a další task provede nakopírovaní potřebných souborů na příslušné místo.

Pro instalaci balíčků grunt preprocesoru less a balíčku, který se stará o kopírování souborů, je nutné spustit následující příkazy:

```
> npm install grunt-contrib-less --save-dev
> npm install grunt-contrib-copy --save-dev
```

Poté je třeba upravit konfiguraci Gruntfile.js

Ukázka 2 - Úprava konfigurace Gruntfile.js

```
module.exports = function(grunt) {
   grunt.initConfig({
      //definování tasku pro vygenerování css souboru z less
      less: {
         dist: {
            files: {
               'www/css/test.css': 'tpl/less/test.less'
            }
         }
      },
      //definování tasku pro vykopírování knihovny jquery do složky www
      copy: {
         main: {
            files: [
               {expand: true, cwd: 'node_modules/jquery/dist/',  src:
['jquery.js'], dest: 'www/js/'},

            ],
         },
      },
   });
```

```
    //Načtení potřebných balíčků
    grunt.loadNpmTasks('grunt-contrib-less');
    grunt.loadNpmTasks('grunt-contrib-copy');

    //definování defaultního tasku
    //spuštění: grunt
    grunt.registerTask('default', ['copy', 'less']);

    //definování vlastního tasku
    //spuštění: grunt css
    grunt.registerTask('css', ['less']);

};
```

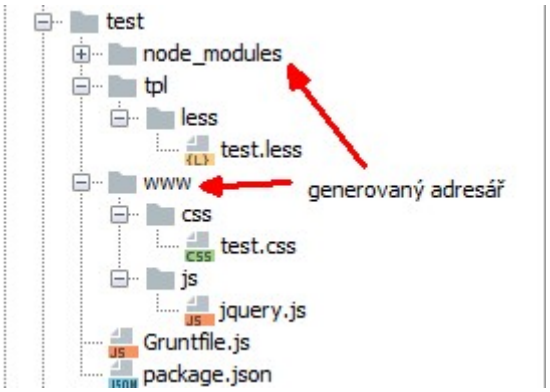
Spuštění Grunt tasků je už pak velmi jednoduché. Pokud je definován ‚default' task, tak lze spustit jednoduše pomocí příkazu:

```
> grunt
```

Lze též definovat pomocí parametru konkrétní task. Například tento příkaz spustí task pro vygenerování less souborů do css:

```
> grunt css
```

Výsledná struktura projektu je uvedena na následujícím obrázku. Složky ‚node_modules' a ‚www' jsou generovány ze zdrojových souborů.



Obrázek 11 - Příklad struktury projektu v Gruntu – zdroj: autor

4.1.2.2 Shrnutí

Výše uvedený přiklad implementace ukazuje, že možností, jakým způsobem lze definovat development workflow pro kodéry, je opravdu mnoho. Důležité je, že tyto nástroje existují a umožňují zefektivnit práci kodérům. Tvorba obecného standardu záleží na konkrétním kodérovi nebo na domluvě v rámci developerského týmu. Jak bylo zmíněno výše, nástroje

pro frontend v tomto ohledu dávají vývojářům velkou svobodu a vždy záleží na zkušenostech a praxi člověka, který nastaví tato pravidla.
Díky task manažerům lze vytvářet HTML šablony velmi efektivně, a hlavně se dají jednoduše svěřit programátorům, kteří budou dané šablony oživovat (33). Zdrojové kódy (less, sass, coffescript atd.) mohou buď napojit na vlastní task manažer, nebo použijí již hotovou konfiguraci od kodéra a implementují ji do oživovaného projektu.
Task manažery hrají také velkou roli i při samotném nasazení (deployment), protože se dají nadefinovat tasky pro vývojářskou stanici a tasky pro build proces přímo na serveru (36). Node.js je serverový jazyk, takže lze tyto task manažery na serveru spustit a při přípravě build procesu nastavit příslušné grunt tasky.

4.1.3 Výstup práce kodéra na tvorbě HTML šablon

Jak již bylo zmíněno předchozích podkapitolách, kodér dodává nejen HTML šablony (zdrojové soubory preprocesoru css, zdrojové soubory javasciptu, zdrojové soubory jednotlivých HTML šablon), ale i konfiguraci pro build šablon ze zdrojových souborů. Kodér může předat HTML šablony v zazipovaném souboru, nicméně tato varianta není příliš efektivní, protože při případné opravě se musí poslat další verze souborů. Aby se zamezilo tomuto nepraktickému způsobu předání, je velmi vhodné využívat některý z verzovacích systémů např. GIT. Vývojáři, kteří se starají o oživení HTML šablon, tak mohou mít stále k dispozici aktuální verzi HTML šablon, případně mohou daný repozitář navázat přímo na aktuální projekt.

4.2 Pro MVC Backend frameworky

Níže uvedené frameworky jsou jen výběrem a slouží pouze k získání základní představy, jakým způsobem se do nich implementují dodané HTML šablony. Jak bude níže uvedeno, HTML šablony od kodéra se pouze doplní o proměnné daného šablonovacího systému. Oživení tak v zásadě probíhá pomocí vykopírování zdrojového kódu statické šablony do vytvořené šablony v backend frameworku, který se postará o naplnění požadovanými daty. Vzhled a chování na frontendu se pak již řídí pomocí .css a .js souborů, které jsou vygenerovány pomocí některého task manažeru.

4.2.1 Nette

Jedná se o velmi rozšířený PHP framework zvláště v České Republice, kde ho používají známé projekty a firmy: ulož.to, Internet Info, VltavaLabePress, Slevomat a mnoho dalších. (37)

Nette používá šablonovací systém Latte, který je velmi přehledný a snadno se s ním pracuje, takže oživení HTML šablon může po krátkém zaučení zvládnout i sám kodér. Latte obsahuje celou řadu maker a filtrů.

Ukázka 3 - Příklad Latte šablony – HTML

```
<div class="main-content">
   <div class="article-detail-description">
      <div class="wsw">
         <p>Hello, {$name|noescape} !</p>
      </div>
   </div>
</div>

</div>
```

4.2.2 Spring MVC

Spring je MVC framework postavený nad programovacím jazykem JAVA (38). Používá se pro tvorbu JAVA aplikací. V rámci Spring frameworku se nejčastěji používá na vykreslení frontendu šablony JSP. Oproti Latte šablonám JSP vyžadují důkladnější seznámení se syntaxí. Nicméně stále platí, že oživení dodaných HTML šablon není nijak složité.

Ukázka 4 - Příklad JSP šablony – HTML

```
<div class="main-content">
   <div class="article-detail-description">
      <div class="wsw">
         <p th:text="'Hello, ' + ${name} + ' !'" />
      </div>
   </div>
</div>
```

4.2.3 ASP.NET

ASP.NET je open source platforma napsaná v .NET. Používá se pro tvorbu webových stránek. Zaměřuje se na trendy ve vývoji frontendu a nabízí řadu služeb, které usnadňují vývojářům této platformy rychleji vyvíjet požadované aplikace (39). Šablonovací systém ASP.NET nebrání nijak při implementaci dodaných šablon od kodéra.

Ukázka 5 - Příklad ASP.NET šablony – HTML

```
<div class="main-content">
    <div class="article-detail-description">
        <div class="wsw">
            <p>Hello, @name !</p>
        </div>
    </div>
</div>
```

4.2.4 Shrnutí

Jelikož výše uvedené frameworky žádným způsobem nelimitují použité knihovny javascriptu, popř. případné CSS frontend frameworky, jsou velmi dobře připravené na implementaci jakýchkoli HTML šablon. Nemají problém s implementací task manažeru, který provede build souborů potřebných pro zobrazení frontendu v prohlížečích.

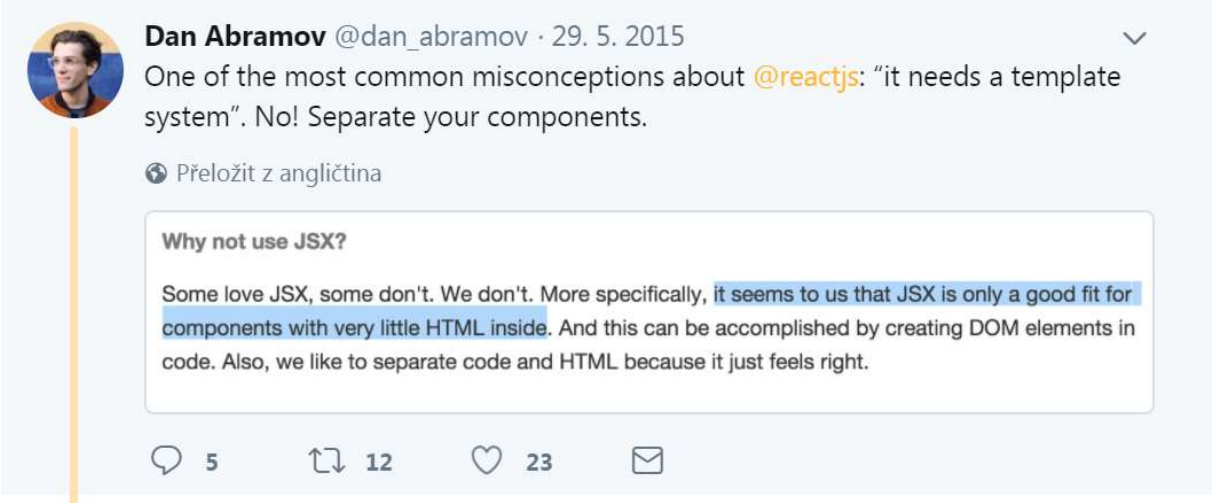
4.3 Pro Javascriptové frameworky/knihovny

V případě implementace HTML šablon do šablonovacích systémů javascriptových frameworků/knihoven už vznikají jisté potíže, které se bude snažit postihnout praktická část této práce. Jedná se především o nekompatibility nástrojů použitých kodérem pro build HTML šablon. Dále je problém v použitých knihovnách javascriptu, které mohou fungovat např. nad knihovnou jQuery a nebudou kompatibilní s použitým javascript frameworkem. V neposlední řadě je HTML syntaxe, která může být odlišná proti standardnímu HTML – např. React.

4.3.1 React

React používá jako šablonovací systém JSX. Jedná o systém, který vypadá velmi podobně jako HTML, nicméně se následně při buildu přegenerovává na javascriptové funkce, což znamená, že při oživování se musí dodané šablony od kodéra přepsat do JSX a následně oživit (40). V případě, že kodér vytvořil šablony nad CSS frontend frameworkem, který používá jQuery, museli by programátoři Reactu složitě přenášet a bindovat na komponenty Reactu. V zásadě tento postup nedoporučuje ani samotná komunita React vývojářů (41). Existuje projekt react-templates, který umožňuje vytvořit samostatný šablonovací systém i v rámci Reactu (42). To znamená, že vývojáři mají k dispozici soubory šablon

podobně jako u MVC projektů. Nicméně podle názoru respektovaného odborníka Dana Abromova se jedná o nepochopení konceptu celého Reactu. (43)



Obrázek 12 - Dan Abramov – Twitter – zdroj: (43)

Ukázka 6 - Příklad JSX šablony

```
const element = (
   <div>
      <div className="main-content">
         <div className="article-detail-description">
            <div className="wsw">
               Hello, {formatName(name)} !
            </div>
         </div>
      </div>
   </div>
);

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

4.3.2 Angular 2

Angular, na rozdíl od Reactu, umožňuje mít samostatnou HTML šablonu. Používá i standardní koncovku u souboru .html. Nicméně Angular umožňuje definovat i template v rámci konfiguračního souboru komponenty. Co se týče implementace javascriptu v rámci dodávky HTML šablon, je na tom hodně podobně jako React. Nějakým způsobem lze dostat jiné javascriptové knihovny do Angular aplikace, nicméně to není dobrý nápad z hlediska udržitelnosti. (44)

Ukázka 7 - Příklad načtení externí HTML šablony

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: [ './app.component.css' ]
})
```

Ukázka 8 - Obsah externí HTML šablony

```
<div class="main-content">
   <div class="article-detail-description">
      <div class="wsw">
         <p>Hello, {{hero.name}} !</p>
      </div>
   </div>
</div>
```

Ukázka 9 - Příklad šablony definované přímo v .ts

```
@Component({
  selector: 'app-root',
  styleUrls: [ './app.component.css']
  template: <div class="main-content">
   <div class="article-detail-description">
      <div class="wsw">
         <p>Hello, {{hero.name}} !</p>
      </div>
   </div>
</div>`
})
```

4.3.3 Shrnutí

Jak bylo zmíněno hned na začátků této podkapitoly, implementace dodaných HTML šablon do javascriptového frameworku/knihovny, je rozhodně složitější než implementace do backendových šablonovacích systémů. Problémem je především práce s javascriptem na straně připravených HTML šablon. Pokud je vůbec nějaký připraven, bývá pro javascript programátory určitě výhodnější vše přepsat do platformy, na které je stavěna aplikace. Potíží se může také stát build samotného frontendu, který s nejvyšší pravděpodobností bude muset být převeden na build dané platformy. Z toho vyplývá, že pokud kodér připravuje šablony javascriptový framework/knihovnu, je důležité, aby od samého začátku spolupracoval s vývojáři aplikace.

5 Praktická část

Praktická část práce má za cíl vyzkoušet implementaci standardní HTML šablony vytvořené pomocí dvou nejpoužívanějších CSS frontend frameworků (Bootstrap, Foundation) do zmíněných javascriptových frameworků/knihoven (React, Angular). Cílem je nasimulovat situaci, kdy frontend developer (kodér) dodá HTML šablony javascript programátorům vytvořené standardní cestou. Na konci praktické části bude provedeno zhodnocení. V rámci něj by se měl čtenář dozvědět:

- Jaká jsou úskalí implementace HTML šablony vytvořené pomocí CSS frontend frameworku do javascriptového frameworku/knihovny
- Návrhy na úpravu vývojového procesu při tvorbě statických HTML šablon, pokud se mají implementovat do javascriptového frameworku/knihovny
- Požadavky na dovednosti kodérů připravující HTML šablony pro javascriptový framework/knihovnu

Praktická část je rozdělena do tří základních kapitol:

- Příprava na tvorbu statických HTML šablon
- Implementace připravených šablon do React a Angular aplikace
- Zhodnocení praktické části

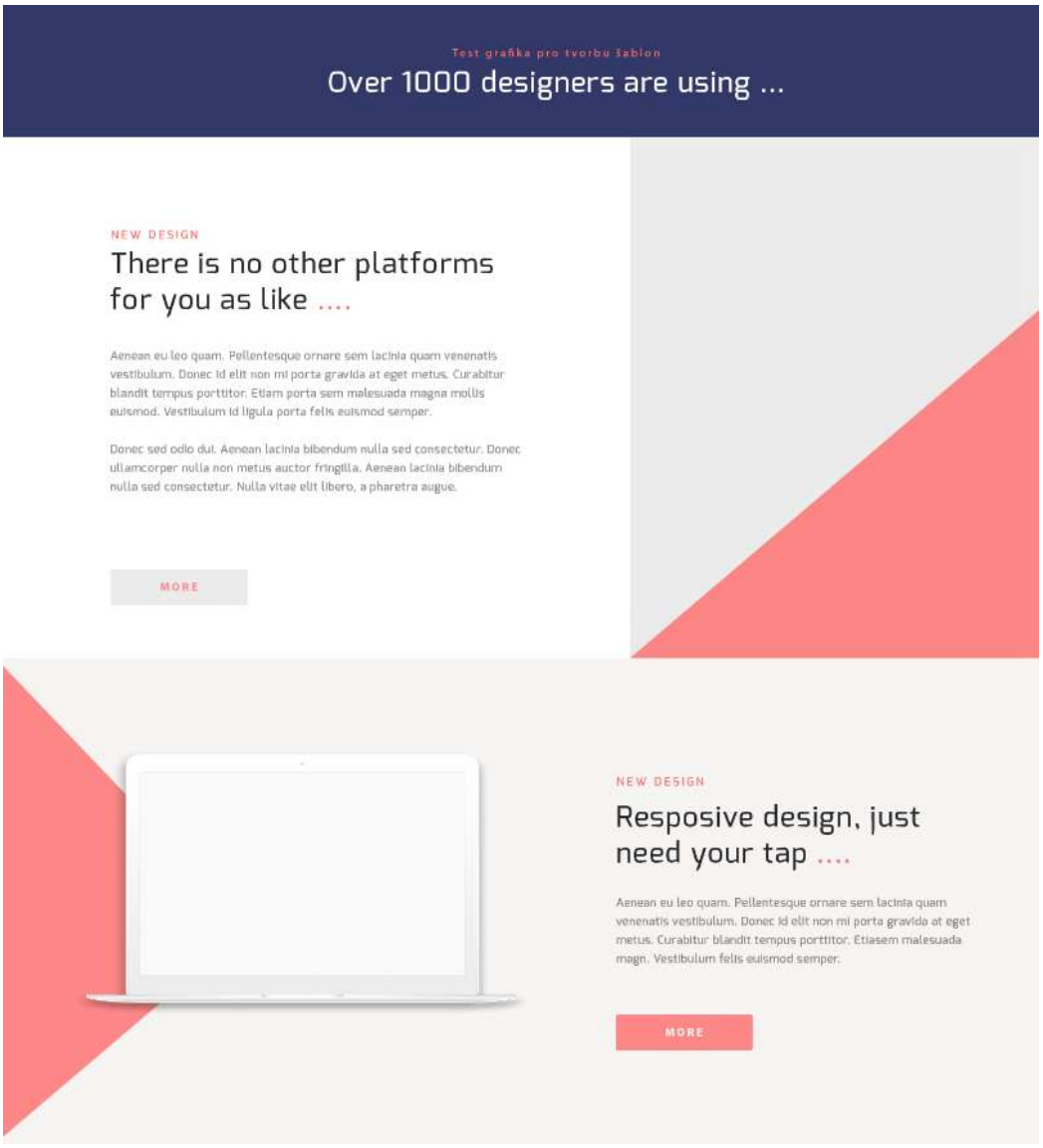
První kapitola se zabývá definicí grafického návrhu, přípravou zadání pro HTML šablonu a samotnému nakódování/přípravě HTML šablony za pomocí Bootstrapu a Foundation. Druhá kapitola se věnuje tvorbě jednoduchých aplikací postavených nad Reactem a Angularem. Následně se provede základní oživení HTML šablony v rámci daného frameworku.

5.1 Příprava na tvorbu statických HTML šablon

Jak již bylo zmíněno v teoretické části práce, každá práce kodéra začíná tím, že dostane grafiku, kterou musí převést do HTML, CSS a Javascriptu. Grafika pro účely praktické části byla vytvořena záměrně jednoduše s několika základními prvky, které se na webových stránkách obvykle objevují.

5.1.1 Grafický návrh šablony

Navržená grafika počítá s responsivním zobrazením šablony. Zachycuje chování šablony na desktopovém zařízení a na mobilním zařízení. Grafika počítá s použitím CSS frontend frameworků. Neobsahuje tedy žádné speciální prvky, které se vymykají běžnému toku obsahu. Grafika je k dispozici v přiloženém .psd souboru.



Obrázek 13 - Desktop verze – převzato z (45) - upraveno autorem

Obrázek 14 - Mobilní verze – převzato z (45) - upraveno autorem

5.1.2 Zadání pro tvorbu HTML šablony

Grafika šablony již v zásadě hodně napoví o tom, jakým způsobem se má obsah na webu chovat a jaký je jeho účel. Nicméně existují samozřejmě i složitější projekty a grafiky, kde je nutné specifikovat chování nejen šablon, ale i komponent, které se v nich vyskytují.

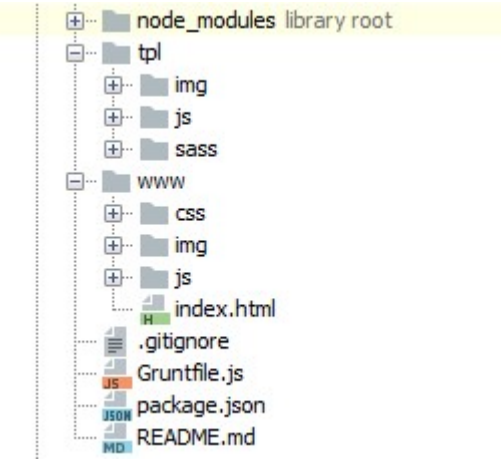
Zadání pro tvorbu požadované šablony je následující. HTML šablona bude nakódována pomocí CSS frontend frameworku, bude responsivní a bude se korektně zobrazovat správně v aktuálních verzích webových prohlížečů IE, Chrome, Firefox. Po kliknutí na tlačítko „more“, se provede rozbalení textu. Pokud se na tlačítko klikne znovu, dojde k zabalení rozbaleného textu.

5.1.3 Příprava vývojového prostředí

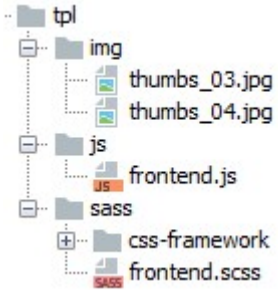
Jak bylo zmíněno v kapitole 4.1., frontend developer používá základní nastavení vývojového prostředí. Používá node.js/npm, task manažer a případně nějaký CSS frontend framework. Vzhledem k tomu, že je úkolem vytvořit HTML šablony nad dvěma CSS frontend frameworky, je jasné, že se dá používat stejná konfigurace vývojového prostředí pro oba projekty. Pouze se v rámci projektu nastaví příslušný CSS frontend framework.

Projekt tedy bude postaven nad npm, Grunt a CSS frontend frameworkem.

Ukázka 10 - Struktura projektu pro tvorbu statických šablon



Ukázka 11 - Složka ./tpl bude obsahovat zdrojové soubory CSS, JS a obrázků (img)



Vzhledem k tomu, že oba zamýšlené CSS frontend frameworky používají CSS preprocesor SASS, lze nastavit do základní konfigurace package.json balíček, který se postará o generování CSS souborů (grunt-sass). Následně se také dá připravit složka, kam se budou soubory CSS preprocesoru ukládat (./tpl/sass/)

Soubor ./tpl/sass/frontend.scss je hlavním souborem pro načtení všech závislostí CSS. V rámci něj se bude definovat dotažení stylů z příslušného CSS frontend frameworku. Pro generování příslušných javascriptů se použije „browserify“ s tím, že hlavním souborem, kde se bude provádět import příslušných javascriptů a knihoven, bude pro soubor ./tpl/js/frontend.js.

Obsah složky ./www je generovaný pomocí Grunt tasků a také se v něm bude nacházet soubor index.html, který bude obsahovat HTML zdrojový kód šablony.

Uložená konfigurace v souboru package.json se postará o dotažení všech potřebných modulů, které jsou nutné pro fungování Gruntu a generování příslušných stylů

a javascriptů daného CSS frontend frameworku. Grunt se ještě bude starat o nakopírování zdrojových obrázků ./tpl/img do složky ./www a také se postará o zmenšení velikosti souboru vygenerovaného javascriptu.

Ukázka 12 - Příklad obsahu souboru package.json bez závislostí na konkrétní CSS frontend framework

```
{
  "name": "nazev-projektu",
  "version": "1.0.0",
  "description": "Template with CSS frontend framework",
  "main": "Gruntfile.js",
  "author": "Marie Vinopal Boštíková",
  "license": "ISC",
  "devDependencies": {
    "grunt": "^1.0.2",
    "grunt-browserify": "^5.2.0",
    "grunt-contrib-copy": "^1.0.0",
    "grunt-contrib-uglify": "^3.3.0",
    "grunt-sass": "^2.1.0"
  }
}
```

Ukázka 13 - Úryvky z obsahu souboru Gruntfile.js

- Definice, která zajišťuje generování CSS stylů z preprocesoru SASS.

```
sass: {
   options: {
      sourceMap: true,
      outputStyle: 'compressed'
   },
   dist: {
      files: {
         'www/css/frontend.css': 'tpl/sass/frontend.scss'
      }
   }
},
```

- Ukázka základního grunt tasku, který se postará o spuštění všech definovaných tasků. To znamená, že po jeho spuštění dojde k nakopírování obrázků (copy) a vygenerování CSS(sass) i javascriptů (browserify, uglify) na příslušná místa.

```
//definování DEFAULT tasku
//spuštění: grunt
grunt.registerTask('default', ['copy','sass','browserify','uglify']);
```

5.2 Šablona pomocí frameworku Bootstrap

CSS frontend framework Bootstrap je velmi rozšířený a populární, proto byl vybrán pro praktickou část, kdy se pomocí něho vytvoří statická šablona. Projekt je uložen ve složce pod názvem: html-template-bootstrap.

5.2.1 Konfigurace vývojového prostředí

Vývojové prostředí bude vycházet z výše popsaného vývojového prostředí. To znamená, že následná ukázka package.json popisuje implementaci závislosti Bootstrapu do projektu.

Ukázka 14 - Rozšíření package.json o Bootstrap a další závislosti (červeně vyznačeno)

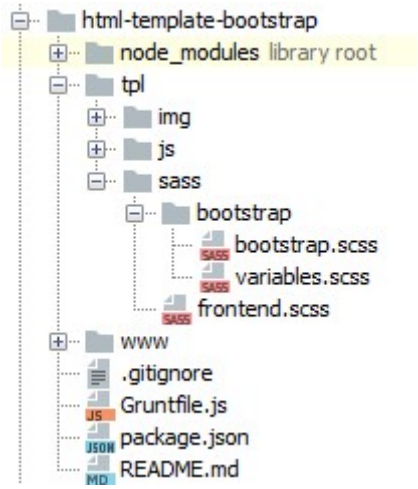
```
{
  "name": "html-template-bootstrap",
  "version": "1.0.0",
  "description": "Simply template with bootstrap CSS frontend framework",
  "main": "Gruntfile.js",
  "author": "Marie Vinopal Boštíková",
  "license": "ISC",
  "devDependencies": {
    "bootstrap": "^4.0.0",
    "grunt": "^1.0.2",
    "grunt-browserify": "^5.2.0",
    "grunt-contrib-copy": "^1.0.0",
    "grunt-contrib-uglify": "^3.3.0",
    "grunt-sass": "^2.1.0",
    "jquery": "^3.3.1",
    "popper": "^1.0.1",
    "popper.js": "^1.14.3"
  }
}
```

5.2.2 Navázání Bootstrap CSS

Pro načtení Bootstrap CSS se použije již připravený soubor ./tpl/sass/frontend.scss, kde dojde k importu vlastní konfigurace Bootstrapu. Výhodou využití zdrojových souborů Bootstrapu je možnost dotažení pouze používaných stylů a možnost nastavení vlastních barev, velikosti písma atd.

V následující části je popsáno, jakým způsobem probíhá konfigurace Bootstrapu na úrovni projektu.

Ukázka 15 - Umístění SCSS souboru Bootstrap šablony



./tpl/sass/bootstrap/frontend.scss - zde se definuje CSS pro HTML šablonu a import vlastní konfigurace Bootstrapu.

Ukázka 16 - Ukázka ze souboru frontend.scss

```
//load bootstrap
@import "bootstrap/bootstrap";

header  {
    text-align:center;
    padding: 15px;
    background: #323969;
    h1 {
        color: #ff6d6d;
        font-size: 16px;
        font-weight: normal;
        margin: 20px 0 10px;
    }
    p {
        font-size: 20px;
        color: #fff;
        @media only screen and (min-width: 768px) {
            font-size: 40px;
        }
    }
}
```

./tpl/sass/bootstrap/bootstrap.scss – zde se provádí import příslušných SCSS pro příslušné komponenty Bootstrapu.

Ukázka 17 - Ukázka ze souboru bootstrap.scss

```
@import "../../../node_modules/bootstrap/scss/functions";
@import "variables"; //ready to change own variables
@import "../../../node_modules/bootstrap/scss/mixins";
@import "../../../node_modules/bootstrap/scss/root";
//@import "../../../node_modules/bootstrap/scss/forms";
@import "../../../node_modules/bootstrap/scss/buttons";
@import "../../../node_modules/bootstrap/scss/transitions";
//@import "../../../node_modules/bootstrap/scss/dropdown";
//@import "../../../node_modules/bootstrap/scss/button-group";
//@import "../../../node_modules/bootstrap/scss/input-group";
```

./tpl/sass/bootstrap/variables.scss – zde se provádí konfigurace proměnných hodnot (např. definice barev, definice velikostí breakpointů atd.).

Ukázka 18 - Ukázky ze souboru bootstrap.scss

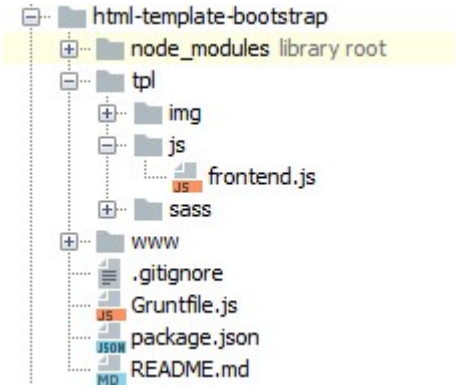
```
// Color system
$primary:       #fc8587 !default;
$secondary:     $gray-600 !default;

// Grid breakpoints
$grid-breakpoints: (
      xs: 0,
      sm: 576px,
      md: 768px,
      lg: 992px,
      xl: 1200px
) !default;
```

5.2.3 Navázání Bootstrap JS

Aby komponenty, které vyžadují javascripty Bootstrapu, správně fungovaly, je nutné zajistit načtení javascriptu z projektu Bootstrap uloženého ve složce „node_modules“. Vzhledem k tomu, že při vývoji HTML standardně dochází k tomu, že javascripty frameworku nedokáží vyřešit požadovanou funkcionalitu, je nutné napsat vlastní javascript. Proto je potřeba zajistit generování těchto javascriptů i javascriptů frameworku. Browserify tyto javascripty spojí dohromady a zajistí přehlednost kódu. Bootstrap javascripty jsou založeny nad knihovnou jQuery. (46)

Ukázka 19 – Umístění zdrojového .js souboru Bootstrap šablony



./tpl/js/frontend.js – zde se provádí import vlastních či externích javascriptových knihoven. Funkcionalita v šabloně (zobrazování/schování obsahu) je velmi jednoduchá a není ani třeba psát vlastní javascript, protože se o to postará již napsaný javascript Bootstrap frameworku.

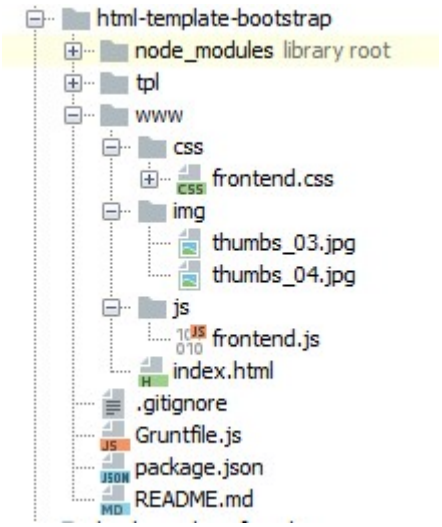
Ukázka 20 - Ukázka ze souboru frontend.js

```
$ = require('jquery');
//load bootstrap javascripts
require('bootstrap');
console.log('Load javascript for frontend')
```

5.2.4 Nakódování HTML šablony pomocí Bootstrapu

V situaci, kdy CSS i javascripty se již povedlo zprovoznit, zbývá to nejdůležitější. Napsání HTML zdrojového kódu a napsání vlastního CSS, případně javascriptů, aby se docílilo požadovaného vzhledu HTML šablony. Samotný zdrojový kód se píše do souboru ./www/index.html a ten používá obrázky, externí javascripty a css soubory, o jejichž generování a umístění se stará grunt. Soubory se nachází v ./www/js/frontend.js, ./www/css/frontend.js a v ./www/img/.

Ukázka 21 – Adresářová struktura souborů .css a .js Bootstrap šablony vygenerovaných pomocí Gruntu



Ukázka 22 - Ukázka ze souboru index.html – vložení stylů

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1,
shrink-to-fit=no">
<!-- Frontend CSS -->
<link href="css/frontend.css" rel="stylesheet">
```

Ukázka 23 - Ukázka ze souboru index.html – vložení javascriptu

```
<!-- Frontend JAVASCRIPT -->
<script src="js/frontend.js"></script>
</body>
</html>
```

Ukázka 24 - Ukázka ze souboru index.html – vložení HTML vycházejícího z Bootstrap – Grid (47)

```
<article role="article" class="row article">
   <div class="article-thumb col-12 col-sm-7"><img src="img/thumbs_04.jpg"
alt="thumb img"/></div>
   <div class="article-content col-12 col-sm-5">
      <p class="tag">New Design</p>
```

Bootstrap umožňuje pomocí speciálních atributů na elementu odřídit chování javascriptu (data-toggle="collapse" data-target="#collapseExample2"). V tomto případě se stará o zobrazení/skrytí části obsahu.

Ukázka 25 - Ukázka ze souboru index.html – vložení HTML vycházejícího z Bootstrap – Collapse (48)

```
<div class="article-summary">
   <p>Aenean ... augue</p>
   <div class="collapse" id="collapseExample2">
      <div>
         <div class="well">
            <p>Aenean…
               augue.</p></div>
      </div>
   </div>
</div>
<button class="btn btn-primary" type="button" data-toggle="collapse" data-
target="#collapseExample2"
        aria-expanded="false" aria-controls="collapseExample2">
   More
</button>
```

HTML šablona je nakódována nad Bootstrap frameworkem a je připravena k implementaci na javascriptový framework.

5.3 Šablona pomocí frameworku Foundation

Při vývoji šablon nad Foundation se bude díky přednastavenému vývojovému prostředí pracovat velmi podobně. Hlavní rozdíly se nachází v implementaci CSS a implementaci jeho tříd do HTML. Foundation má svoje javascripty pro komponenty napsané také pomocí jQuery. Projekt je uložen ve složce pod názvem: html-template-foundation.

5.3.1 Konfigurace vývojového prostředí

Příslušný package.json popisuje implementaci závislosti na Foundation projektu. Vzhledem k tomu, že Foundation používá pro psaní javascriptu ECMAScript 2015, je nutné stáhnout i balíčky pro kompilaci javascriptu foundation pomocí browserify (babel atd.) (48)

Ukázka 26 - Rozšíření package.json o Foundation a další závislosti (červeně)

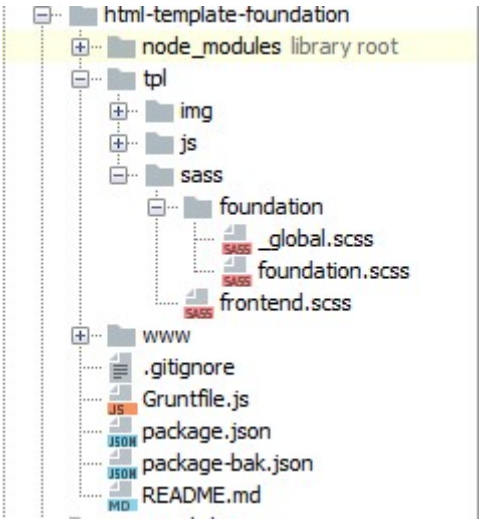
```
{
  "name": "html-template-foundation",
  "version": "1.0.0",
  "description": "Simply template with foundation CSS frontend framework",
  "main": "Gruntfile.js",
  "author": "Marie Vinopal Boštíková",
  "license": "ISC",
  "devDependencies": {
    "babel-core": "^6.26.0",
    "babel-preset-es2015": "^6.24.1",
```

```
    "babelify": "^8.0.0",
    "foundation-sites": "^6.4.4-rc1",
    "grunt": "^1.0.2",
    "grunt-browserify": "^5.2.0",
    "grunt-contrib-copy": "^1.0.0",
    "grunt-contrib-uglify": "^3.3.0",
    "grunt-sass": "^2.1.0"
  }
}
```

5.3.2 Navázání Foundation CSS

Napojení Foundation zdrojových souborů probíhá podobně jako při implementaci Bootstrapu. Samozřejmě je rozdíl v obsahu souborů, které se starají o specifické nastavení pro daný projekt.

Ukázka 27 - Umístění SCSS souboru Foundation šablony



./tpl/sass/foundation/frontend.scss - zde se definuje CSS pro HTML šablonu a import vlastní konfigurace Foundation

Ukázka 28 - Ukázka ze souboru frontend.scss

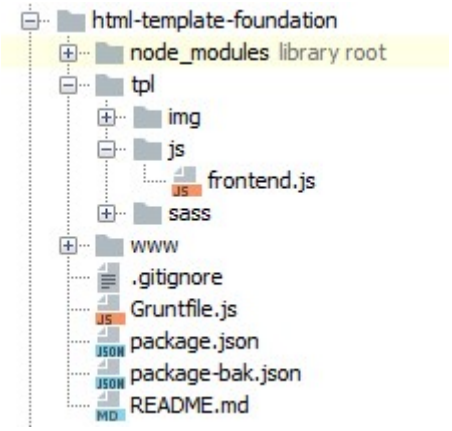
```
//load foundation - own configuration
@import "foundation/foundation";
@include foundation-everything;

.collapse {
   display: none;
   &.expanded {
      display: block;
   }
}
```

5.3.3 Navázání Foundation JS

Foundation podobně jako Bootstrap nabízí řadu předpřipravených komponent, které společně s Javascriptem vytváří chování webové stránky. Samotné navázání probíhá v podstatě stejně jako u Bootstrapu. Pomocí browserify se dotáhnou příslušné závislosti na Foundation frameworku uloženého ve složce „node_modules“.

Ukázka 29 - Umístění zdrojového .js souboru Foundation šablony



./tpl/js/frontend.js

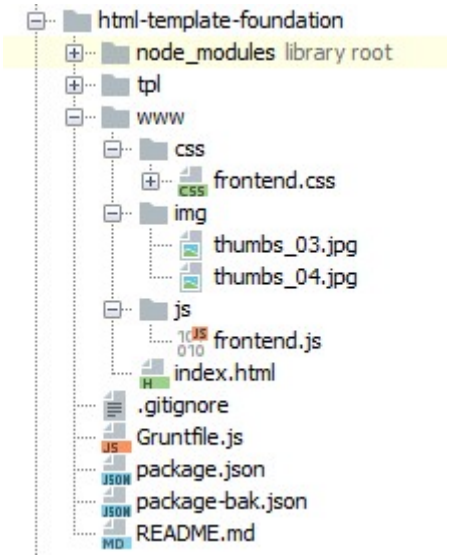
Ukázka ze souboru frontend.js

```
window.$ = window.jQuery = require('jquery');
//load bootstrap javascripts
require('foundation-sites');
$(document).foundation();
console.log('Load javascript for frontend')
```

5.3.4 Nakódování HTML šablony pomocí Foundation

Vývojové prostředí po téměř identických krocích umí generovat CSS a Javascripty pro Foundation. To znamená, že se může začít s přípravou konkrétního HTML šablony. Vzhledem k tomu, že se jedná o velmi jednoduchou HTML šablonu, tak se přepoužije HTML kód, který se připravil pro Bootstrap. Provede se úprava na úrovni elementů, které používaly logiku Bootstrapu (grid, button, rozbalení/zabalení obsahu – javascript). Struktura projektu zůstává stejná jako u Bootstrapu, takže lze použít provázání css a javascriptu v HTML šabloně.

Ukázka 30 - Adresářová struktura souborů .css a .js Foundation šablony vygenerovaných pomocí Gruntu



Ukázka 31 - Ukázka ze souboru index.html – vložení HTML vycházejícího z Foundation – Grid (49)

```
<article role="article" class="grid-x grid-padding-x  article">
   <div class="article-thumb  cell small-12 medium-7"><img
src="img/thumbs_04.jpg" alt="thumb img"/></div>
   <div class="article-content  cell small-12 medium-5">
      <p class="tag">New Design</p>
```

Foundation poskytuje řadu komponent s příslušnými javascripty, stejně jako ostatní CSS frontend frameworky. Pro odřízení (ne)zobrazení obsahu po kliknutí na tlačítko „more“ se používá komponenta Toggler. Co se týče stylování tlačítek, tak je velmi podobné tomu, jak se používají v Bootstrapu.

Ukázka 32 - Ukázka ze souboru index.html – vložení HTML vycházejícího z Foundation – Toggler (50)

```
<div class="article-summary">
   <p>Aenean   augue.</p>
   <div class="collapse" id="collapseExample1" data-toggler=".expanded"
aria-expanded="false">
      <div>
         <div class="well">
            <p>Aenea augue.</p></div>
      </div>
   </div>
</div>
<button class="button primary" type="button" data-toggle="collapseExample1"
aria-expanded="false" aria-controls="collapseExample1">
   More
</button>
```

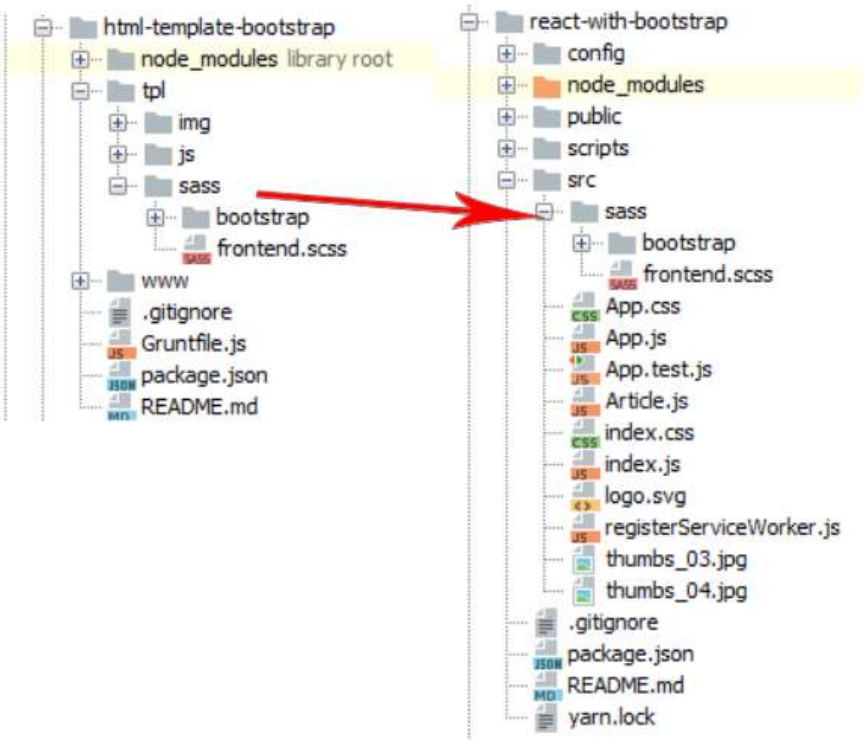
5.4 Implementace šablon do Reactu

Jak bylo zmíněno v úvodu praktické části, React aplikace bude velmi jednoduchá a bude postavena na základě startovacího projektu od React vývojářů. (51) Po instalaci je ještě nutné provést tzv. eject, který v rámci projektu vygeneruje všechny soubory včetně konfiguračních, což je v rámci tohoto projektu potřeba. Protože je třeba konfigurovat webpack pro SCSS. Cílem implementace je zobrazení HTML stránky vykreslené pomocí Reactu. React neumožňuje oddělení HTML šablony do samostatného souboru. Definování HTML šablony v React komponentě je na jednom místě s logikou zobrazování šablony a dalšími konfiguracemi např. nastavení metod, které odřídí chování po kliknutí na příslušný element. Projekty jsou uloženy ve složkách pod názvy: react-with-bootstrap, react-with-foundation.

5.4.1 Struktura projektu

V souborech App.js a Article.js probíhá implementace dodané HTML šablony.
V příslušném React projektu (Bootstrap, Foundation) je složka ./src/sass/, kam se ručně nakopírovaly soubory ze statické šablony (./tpl/sass/).

Ukázka 33 – Implementace zdrojových .scss souborů do React projektu



5.4.2 Konfigurace Webpack – SCSS

React nepoužívá Grunt, ale Webpack, který se stará o build celé aplikace. Webpack je třeba konfigurovat, aby dokázal generovat dodané stylové soubory SASS (.scss). Vyřeší se přidáním loaderu do souboru konfigurace webpacku, který se stará o vygenerování .scss souborů do css. (52)
Je nutné nainstalovat ještě příslušné npm balíčky.

```
npm install sass-loader node-sass webpack --save-dev
```

Ukázka 34 - Přidání loaderu do ./config/ webpack.config.dev.js

```
{
 test: /\.scss$/,
 loaders: [ 'style-loader', 'css-loader', 'sass-loader' ]
}
```

Implementace stylů do aplikace se řeší pomocí import .scss souboru v rámci souboru App.js.

Ukázka 35 - Úryvek ze zdrojového kódu App.js

```
import React, {Component} from 'react';
import thumb1 from './thumbs_03.jpg';
import thumb2 from './thumbs_04.jpg';

//import css
import './sass/frontend.scss';
```

5.4.3 Bootstrap

V případě, že je React aplikace nakonfigurována dle výše uvedených požadavků, je připravena na implementaci nakódované Bootstrap šablony (html-template-bootstrap).
Nejdříve je třeba do React projektu stáhnout npm balíček pro Bootstrap.

```
npm install bootstrap --save-dev
```

Poté se provede nakopírování SCSS složky z [html-template-bootstrap] šablony na příslušné místo v React projektu. (viz. Struktura projektu). V rámci SCSS souborů je nutné ošetřit cesty do složky „node_modules“
Po zprovoznění buildu Bootstrap stylů aplikace je možné začít se základním oživení React aplikace.

Ukázka 36 - Úryvek z App.js – render metoda na vykreslení obsahu

```
render() {
   return (
      <div className="App">
         <header>
            <div className="container">
               <h1>Test grafika pro tvorbu šablon</h1>
               <p>Over 1000 designers are using ...</p>
            </div>
         </header>
         <main role="main" className="container">
            <Article content={this.article1} />
            <Article content={this.article2} />
         </main>

      </div>
   );
}
```

Ukázka 37 - Úryvek z Article.js – render metoda na vykreslení článku

```
render() {
   // This syntax ensures `this` is bound within handleClick
   return (
      <article role="article" className="row article">
         <div className="article-thumb col-12 col-sm-5 col">
            <img src={this.props.content.thumb} alt="thumb img"/>
         </div>
         <div className="article-content  col-12  col-sm-7 col">
            <p className="tag">New Design</p>
            <h2>{this.props.content.title}</h2>
            <div className="article-summary">
               <p>Aenean eu leo quam. Pe ... lorem</p>
               <div className={this.state.isToggleOn ? 'collapse show' :
'collapse'} id={this.articleId}>
                  <div>
                     <div className="well">
                        <p>Aenean eu leo quam. Pe ... lorem</p>
                     </div>
                  </div>
               </div>
            </div>
            <button onClick={this.handleClick} className="btn btn-primary"
type="button" aria-expanded={this.state.isToggleOn}>
               More
            </button>
         </div>
      </article>
   );
```

Ukázka 38 - Úryvek z Article.js – vyřešení (ne)rozbalení pomocí Reactu

```
class Article extends React.Component {
   constructor(props) {
      super(props);
      this.state = {isToggleOn: false};

      // This binding is necessary to make `this` work in the callback
```

```
        this.handleClick = this.handleClick.bind(this);
    }
    handleClick() {
        this.setState(prevState => ({
            isToggleOn: !prevState.isToggleOn
        }));
    }
```

V případě funkcionality řešené pomocí Bootstrap javascriptu je určitě rychlejší napsat vlastní metodu přímo v Reactu, než složitě řešit import javascriptů Bootstrapu, které jsou navíc postavené nad jQuery.

5.4.4 Foundation

Do React aplikace s výše uvedenou konfigurací stačí dohrát Foundation přes npm a může se začít implementovat dodaná šablona. Pokud se použije projekt z Bootstrapu, tak se v podstatě v rámci logiky Reactu nemusí nic řešit (např. javascript na (ne)zobrazení obsahu pouze potřebuje definovat korektní CSS třídu).

```
npm install foundation-sites --save-dev
```

Foundation styly se vygenerují stejně jako pro Bootstrap projekt a také se stejně importují v rámci App.js U javascriptů v rámci statické HTML šablony je to podobné jako u Bootstrapu. [react-with-bootstrap).

Ukázka 39 - Úryvek z App.js – render metoda na vykreslení obsahu

```
render() {
    return (
<div className="App">
     <div className="App">
         <header>
             <div className="container">
                 <h1>Test grafika pro tvorbu šablon</h1>
                 <p>Over 1000 designers are using ...</p>
             </div>
         </header>
         <main role="main" class="grid-container">
             <Article content={this.article1} />
             <Article content={this.article2} />
         </main>
     </div>
</div>

    );
}
```

Ukázka 40 - Úryvek z Article.js – render metoda na vykreslení článku

```
render() {
   // This syntax ensures `this` is bound within handleClick
   return (
      <article role="article" class="grid-x grid-padding-x article">
   <div class="article-thumb cell small-12 medium-5">
      <img src={this.props.content.thumb} alt="thumb img"/>
   </div>
   <div class="article-content cell small-12 medium-7">
      <p className="tag">New Design</p>
      <h2>{this.props.content.title}</h2>
      <div className="article-summary">
         <p>Aenean </p>
         <div className={this.state.isToggleOn ? 'collapse expanded' :
'collapse'} id={this.articleId}>
            <div>
               <div className="well">
                  <p>Aenean…augue.</p>
               </div>
            </div>
         </div>
      </div>
      <button onClick={this.handleClick} className="button primary"
type="button" aria-expanded={this.state.isToggleOn}>
         More
      </button>
   </div>
</article>
   );
```

Co se týče implementace javascriptů, narazilo se na stejný problém jako u Reactu, kdy implementace javascriptů do projektu zabere více času, než kdyby se napsaly znovu.

5.5 Implementace šablon do Angularu

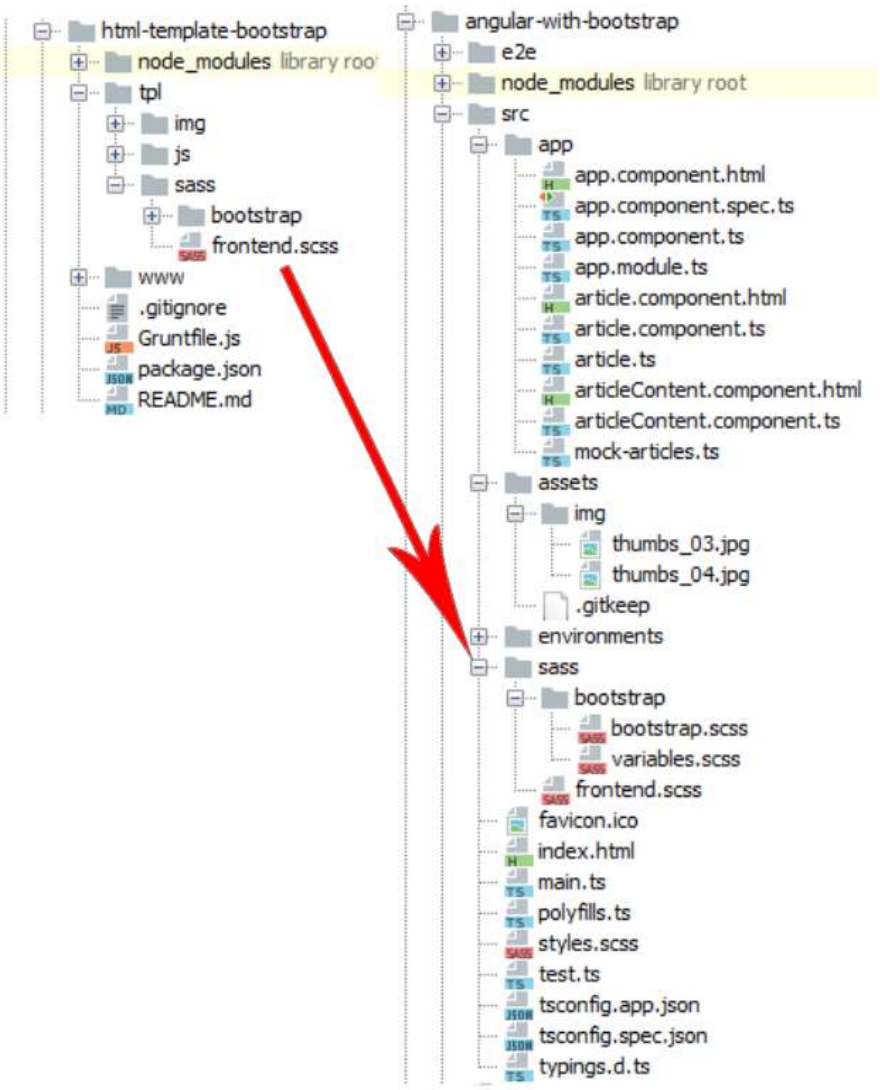
Při implementaci statických HTML šablon do Angular aplikace se může použít obdobný koncept jako při implementaci šablon do React aplikace. Pro přípravu aplikace se použije startovací projekt Angularu, který se nakonfiguruje pro potřeby napojení příslušného CSS frontend frameworku. (53) Jak bylo zmíněno v kapitole 4.3.2, Angular umožňuje, na rozdíl od Reactu, definovat vlastní html šablony v samostatném souboru. V rámci tohoto souboru lze používat standardní HTML. Projekty jsou uloženy ve složkách pod názvy angular-with-bootstrap, angular-with-foundation.

5.5.1 Struktura projektu

Sobory potřebné pro implementaci HTML se nachází ve složce (./src/app/). Soubory mají koncovku klasicky .html. Např.: app.component.html nebo app.articleContent.html.

Zdrojové soubory stylů se nahrají do složky (./src/sass/).

Ukázka 41 - Implementace zdrojových .scss souborů do Angular projektu



5.5.2 Konfigurace Angular aplikace

V konfiguračním souboru Angular aplikace(./.angular-cli.json) se definuje název souboru stylů (./src/app/style.scss), kde se následně provede import požadovaných zdrojových stylů HTML šablony. (54) Ještě je potřeba dohrát pomocí npm balíček na generování SASS souborů. Konfigurace pro generování CSS souborů z preprocesoru v případě angularu nezabere v podstatě žádný čas.

Ukázka 42 - Úryvek z konfiguračního souboru Angular aplikace (./.angular-cli.json)

```
"index": "index.html",
"main": "main.ts",
"styles": [
  "styles.scss"
],
"script
```

Ukázka 43 - Úryvek z ./src/app/styles.scss

```
/* You can add global styles to this file, and also import other style
files */
@import "./sass/frontend.scss";
```

5.5.3 Bootstrap

Pro dotažení závislosti na bootstrapu je nutné v rámci Angular projektu spustit:

```
npm install bootstrap --save-dev
```

Dále je potřeba nahrát zdrojové soubory statické HTML šablony na příslušná místa (obrázky, img, .scss soubory). Co se týče javascriptů, je podobně jako u Reactu rychlejší vyřešit požadovanou funkcionalitu na základě Angularu.

Ukázka 44 - Úryvek z app.component.html šablona na vykreslení obsahu

```
<header>
  <div class="container">
    <h1>{{title}}</h1>
    <p>{{subtitle}}</p>
  </div>
</header>
<main role="main" class="container">
  <article-component></article-component>
</main>
```

Ukázka 45 - Úryvek z articleContent.component.html – šablona, která vykresluje článek

```
<article role="article" class="row article">
  <div class="article-thumb col-12 col-sm-5 col "><img
src="{{article.thumb}}" alt="thumb img"/></div>
  <div class="article-content  col-12  col-sm-7 col">
    <p class="tag">New Design</p>
    <h2>{{article.name}}</h2>
    <div class="article-summary">
      <p>Aenean … augue.</p>
      <div class="collapse" [ngClass]="openMore ? 'show' : ''">
        <div>
          <div class="well">
            <p>Aenean … </p>
          </div>
        </div>
      </div>
```

```
    </div>
    <button class="btn btn-primary" type="button" (click)="onClickMore()"
            aria-expanded="false" aria-controls="collapseExample1">
      More
    </button>
  </div>
</article>
```

5.5.4 Foundation

Foundation se stáhne pomocí npm.

```
npm install foundation-sites --save-dev
```

Následně se nahrají podobně jako v v projektu Angular Bootstrap všechny zdrojové soubory na příslušná místa.

Ukázka 46 - Úryvek z app.component.html šablona na vykreslení obsahu

```
<header>
  <div class="container">
    <h1>{{title}}</h1>
    <p>{{subtitle}}</p>
  </div>
</header>

<main role="main" class="grid-container">
  <article-component></article-component>
</main>
```

Ukázka 47 - Úryvek z articleContent.component.html – šablona, která vykresluje článek

```
<article role="article" class="grid-x grid-padding-x article">
      <div class="article-thumb cell small-12 medium-5"><img
src="{{article.thumb}}" alt="thumb img"/></div>
      <div class="article-content cell small-12 medium-7">
    <p class="tag">New Design</p>
    <h2>{{article.name}}</h2>
    <div class="article-summary">
      <p>Aeneah … augue.</p>
      <div class="collapse" [ngClass]="openMore ? 'expanded' : ''">
        <div>
          <div class="well">
                  <p>Aeneah … augue.</p>
          </div>
        </div>
      </div>
    </div>
    <button class="button primary" type="button" (click)="onClickMore()"
aria-expanded="false">
      More
    </button>
  </div>
</article>
```

5.6 Vyhodnocení

Praktická část potvrdila řadu tvrzení, která zazněla v teoretické části práce. Navíc se povedlo identifikovat i možné změny při práci kodérů na tvorbě HTML šablon pro javascriptové frameworky.

5.6.1 Implementace Bootstrapu do Reactu

Implementace šablony nebyla v případě takto jednoduchého projektu nijak složitá, nicméně pokud by se jednalo o složitější aplikaci a dodržel by se i korektní přístup k tvorbě jednotlivých komponent, tak by určitě začalo dávat smysl řešit html šablony takovým způsobem, aby se snadno daly vkládat do webu. Na internetu existuje ke stažení několik projektů (55), které se zabývají tvorbou „Reactích" komponent postavených nad HTML šablonami Bootstrapu. Tyto projekty nevyvíjí přímo vývojáři Bootstrapu, ale samostatná komunita, která se snaží reflektovat změny, které Bootstrap vydává ve své hlavní verzi. (56)

Použití těchto předpřipravených komponent vypadá na první pohled v pořádku. Nicméně při bližším prozkoumání zjistíme, že příliš nedává smysl, aby kodér musel např. element tlačítka přepisovat do speciálního tagu, který pak bude umět interpretovat vývojář, který zná příslušnou knihovnu, která se stará o vykreslení Bootstrap komponenty. React pak tedy přepisuje standardní HTML tagy CSS frontend frameworku na vlastní „Reactí" komponenty, což vypadá na první pohled docela použitelně, nicméně pokud kodér upravuje HTML Bootstrap komponenty, tak jeho změny implementovat do předpřipravené React Bootstrap komponenty může představovat problém. Navíc pokud připravil funkcionalitu javascriptu navázanou na Bootstrap javascript, tak potom musí programátor Reactu vše předělat do Reactu. Ukázka níže dokazuje, že i jednoduché a primitivní tagy lze nahradit něčím speciálním, čemu pak porozumí pouze zkušený React programátor nebo kodér.

Ukázka 48 - Úryvek render.jsx šablony, která se stará o vykreslení React komponenty – Dropdown (57)

```
<ButtonToolbar>
   <DropdownButton
         bsSize="xsmall"
         title="Extra small button"
         id="dropdown-size-extra-small"
   >
```

```
        <MenuItem eventKey="1">Action</MenuItem>
        <MenuItem eventKey="2">Another action</MenuItem>
        <MenuItem eventKey="3">Something else here</MenuItem>
        <MenuItem divider />
        <MenuItem eventKey="4">Separated link</MenuItem>
    </DropdownButton>
</ButtonToolbar>
```

Ukázka 49 - Úryvek HTML Bootstrap komponenty - Dropdown (58)

```
<div class="dropdown">
    <button class="btn btn-secondary dropdown-toggle" type="button"
id="dropdownMenuButton" data-toggle="dropdown" aria-haspopup="true" aria-
expanded="false">
        Dropdown button
    </button>
    <div class="dropdown-menu" aria-labelledby="dropdownMenuButton">
        <a class="dropdown-item" href="#">Action</a>
        <a class="dropdown-item" href="#">Another action</a>
        <a class="dropdown-item" href="#">Something else here</a>
    </div>
</div>
```

Velmi snadná byla implementace .scss stylů, které bylo možné jednoduše importovat do projektu a webpack se postaral o jejich build. Nastavení otevírání/zavírání obsahu po kliknutí na tlačítko „more“ šlo udělat jednoduše v rámci Reactu. Rozhodně se v praktické části upustilo od implementace javascriptu Bootstrapu, který je postavený nad jQuery.

5.6.2 Implementace Foundation do Reactu

Díky tomu, že se v praktické části vycházelo z jedné šablony i funkcionality, bylo zprovoznění aplikace napojené na Foundation již velmi jednoduché. Nahradily se .scss soubory Bootstrapu za Foundation. Následně se upravil zdrojový kód šablon v komponentách Reactu. Funkcionalita rozbalení/zavření obsahu již byla naprogramována, takže se pouze upravily příslušné CSS třídy, které zajistily zobrazování/skrývání obsahu. Implementace Foundation javascriptů se také neřešila, protože byla zbytečná.

5.6.3 Implementace Bootstrapu do Angularu

Při implementaci Angularu bylo velmi jednoduché definovat HTML šablony pro jednotlivé komponenty v samostatných souborech .html. Oživení šablony hodně připomínalo oživení frontend frameworku v rámci MVC Backendových nástrojů. Šablonovací systém je tedy intuitivnější než „Reactí“ .jsx.

```
<ButtonToolbar>{BUTTONS.map(renderDropdownButton)}</ButtonToolbar>
```

Ukázka 50 - Angular – Úryvek z article.component.html

```
<article-content *ngFor="let article of articles" [article]=article
></article-content>
```

5.6.4 Implementace Foundation do Angularu

I v případě implementace připravené Foundation šablony do Angularu vše proběhlo velmi rychle. Stačilo zaměnit zdroj .scss Foundation souborů a následně upravit CSS třídy v html šablonách Angular komponent.

5.6.5 Shrnutí: Implementace CSS frontend frameworku do javascriptového frameworku

Implementace HTML šablony do javascriptových frameworků byla nad očekávání velmi rychlá a její oživení proběhlo bez žádných velkých problémů. Bohužel se při implementaci nedalo použít nastavení vývojového prostředí, které se definovalo pro tvorbu statických HTML šablon. To znamená, že task manažer Grunt se vůbec neintegroval do React ani Angular projektu. Dalším zklamáním byly javascripty připravené kodérem. Jedná se především o předpřipravené funkcionality k jednotlivým komponentám, které se definují na základě atributů na elementu. V rámci HTML šablony bylo úkolem zařídit rozbalení a zabalení textace po kliknutí na tlačítko „More“. Ani v jednom případě implementace CSS frontend frameworku se nevyplatilo přidávat složitě podporu jQuery do React či Angular projektu. Navíc to vývojáři zmíněných javascriptových frameworků ani nedoporučují.

I když se již zjistilo, že z CSS frontend frameworku lze použít aktivně HTML a CSS, neznamená to, že nemá smysl používat javascripty frameworku v rámci přípravy HTML šablony. Právě naopak může jednoduše a efektivně zajistit základní funkcionalitu statické šablony. Díky tomu dostane implementátor interaktivní šablonu s kompletní funkcionalitou a bude se mu rozhodně lépe navrhovat funkcionalita v React, Angular aplikaci. Co se týče stylů, je určitě dobré, že oba nejznámější CSS frontend frameworky používají stejný CSS preprocesor: SASS. Implementace CSS vrstvy frameworků s vlastními styly je velmi snadná. Na závěr lze dodat, že implementace CSS frontend frameworku do javascriptového frameworku je možná, nicméně je limitována nemožností použit javascripto-

vých knihoven CSS frameworku a také specifickým způsobem psaní šablon (např. React, Vue.js).

5.6.6 Doporučení a principy tvorby HTML šablon pro javascriptové frameworky/knihovny

V případě, že bude kodér dodávat HTML šablony pro projekt postavený nad javascriptovým frameworkem (Angular, React, popř. nějaký další), tak je nutné, aby se před začátkem kódování sešel s některým javascript programátorem a domluvili se společně na dodávce šablon. Dále by se měli kodéři Bootstrapu a Foundation naučit používat Webpack a SASS. Vývojové prostředí nad Gruntem je sice plně funkční, nicméně oba nejznámější JS frameworky používají k buildu frontendu Webpack. V případě, že kodér již ovládá React nebo Angular, tak by měl ideálně nakódovat šablony přímo v jednom ze zmíněných javascriptových frameworků. To odpovídá i na otázku, jestli je možné řešit statickou šablonu mimo projekt. V případě SPA aplikací to moc smysl nedává, protože by statické šablony neobsahovaly javascript, který se používá přímo v aplikaci. Ideální by tedy bylo rozšiřovat a kódovat frontend přímo pod platformou daného frameworku. S tím souvisí další požadavek, aby se v podstatě všichni kodéři dovzdělali v základních principech tvorby šablon u dvou nejznámějších javascriptových projektů (React. Angular).

6 Závěr

Cílem diplomové práce bylo seznámit čtenáře se současnými trendy vývoje frontendu aplikací a jakým způsobem se dá pracovat s šablonami postavenými nad nějakým CSS frontend frameworkem, které se pak budou implementovat do javascriptového frameworku. Dílčím cílem je seznámení se základními principy práce kodéra a javascript programátora a v čem se jejich práce liší. Kodér není javascript programátor, nicméně ovládá nakódování HTML šablony na základě dodané grafiky a následně dokáže nakódovat responsive verzi šablon. Javascript programátor není na druhé straně natolik zběhlý v chování šablon v prohlížeči či na mobilu, ale ovládá javascriptový framework a dokáže danou šablonu oživit. Javascript programátor a kodér spolu budou čím dál tím více spolupracovat a je nutné si uvědomit rozdílnost jejich vnímání potřeb ohledně projektu, na kterém participují. Je důležité, aby si kodéři HTML šablon uvědomili, že technologie se neustále vyvíjejí a znát pouze HTML a CSS již v dnešní době rozhodně nestačí. Navíc, jak bylo zjištěno v praktické části, oblíbené task manažery kodérů (Grunt, Gulp) by měly být nahrazeny Webpackem.

Požadavky na kodéry se neustále zvyšují, nicméně je zde třeba také podotknout, že profese kodéra není nikde aktivně vyučována. Na rozdíl od budoucích programátorů, kteří se učí principy programování a na vysokých školách mají na výběr z mnoha oborů aplikované informatiky, kodéři se musí vše naučit sami, popř. praxí v některé z vývojářských firem. Kodér není programátor a řeší pro programátora mnohdy nezáživné věci (např. barva tlačítka a její posun kousek výše). V případě, že vznikají stále nové a nové projekty, kde se kreslí grafika a vizualizace chování aplikace, je potřeba si uvědomit, že to jsou právě kodéři, kteří se starají o převedení zmíněné grafiky do HTML, CSS a JS. Tato diplomová práce by měla pomoci iniciovat přemýšlení kodérů nad tím, co se s jejich nakódovanou šablonou v SPA aplikaci stane a zdali ji budou schopni nadále vylepšovat.

7 Použité zdroje

7.1 Použitá literatura

1. Carey, Patrick. *New Perspectives on HTML5 and CSS3 comprehensive.* Boston : Cengage Learning, 2016. ISBN 978-1-305-50393-9.

2. WHATWG - FAQ. *WHATWG.* [Online] 2018. [Citace: 4. 1 2018.] https://whatwg.org/faq.

3. Leenheer, Niels. HTML5TEST. *How well does your browser support HTML5?* [Online] 2018. [Citace: 4. 1 2018.] https://html5test.com/results/desktop.html.

4. HTML 5.2. *W3C Recommendation.* [Online] W3C, 14. 12 2017. [Citace: 4. 1 2018.] https://www.w3.org/TR/html5/.

5. Bos, Bert. WHAT IS CSS? *Cascading Style Sheets.* [Online] W3C, 2018. [Citace: 5. 1 2018.] https://www.w3.org/Style/CSS/Overview.en.html.

6. Meyer, Eric A. a Weyl, Estelle. *CSS: The Definitive Guide.* Sebastopol, California : O'Reilly Media, 2017. ISBN 978-1-449-39319-9.

7. Bhaumik, Snig. *Bootstrap Essentials.* Birmingham : Packt Publishing, 2015. ISBN 978-1-78439-517-9.

8. Support for older versions of Internet Explorer ended. *Microsoft.* [Online] 2016. [Citace: 5. 1 2018.] https://www.microsoft.com/en-in/windowsforbusiness/end-of-ie-support.

9. Wodehouse, Carey. The Role of a Front-End Web Developer: Creating User Experience & Interactivity. *Hiring Headquaters.* [Online] UpWork, 2018. [Citace: 9. 4 2018.] https://www.upwork.com/hiring/development/front-end-developer/.

10. Marcotte, Ethan. Responsive Web Design. *A List Apart.* [Online] 25. 5 2010. [Citace: 7. 1 2018.] http://alistapart.com/article/responsive-web-design.

11. Havelka, Petr. *Responsivní webdesign.* Praha : Vysoká škola ekonomická v Praze, Fakulta informatiky a statistiky, 2013. Diplomová práce.

12. Desktop vs Mobile vs Tablet Market Share Worldwide. *statcounter.* [Online] 2018. [Citace: 7. 1 2018.] http://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/worldwide.

13. Frain, Ben. *Responsive Web Design with HTML5 a CSS3.* Birmingham : Packt Publishing, 2015. ISBN 978-1-78439-893-4.

14. Gerchev, Ivaylo. The 5 Most Popular Front-end Frameworks Compared. *SitePoint.* [Online] 27. 2 2018. [Citace: 15. 3 2018.] https://www.sitepoint.com/most-popular-frontend-frameworks-compared/.

15. Arsenault, Cody. Top 10 Front-End Frameworks of 2016. *keycdn.* [Online] 5. 2 2018. [Citace: 15. 3 2018.] https://www.keycdn.com/blog/front-end-frameworks/.

16. Bootstrap. *GetBooststrap.* [Online] 2018. [Citace: 13. 1 2018.] https://getbootstrap.com/.

17. Theme Directory. *Wordpress.* [Online] 2018. [Citace: 10. 2 2018.] https://wordpress.org/themes/search/bootstrap/.

18. Our History. *Zurb.* [Online] 2018. [Citace: 11. 2 2018.] https://zurb.com/history.

19. Monteiro, Fernando. *Learning Single-page Web Application Development.* Birmingham : Packt Publishing, 2014. ISBN 978-1-78355-209-2.

20. Buckler, Craig. Best JavaScript Frameworks, Libraries and Tools to use in 2017. *SitePoint.* [Online] 29. 5 2017. [Citace: 16. 3 2018.] https://www.sitepoint.com/top-javascript-frameworks-libraries-tools-use/.

21. Khachatrian, Arshak. *Getting Started with Polymer.* Birmingham : Packt Publishing, 2016. ISBN 978-1-78588-937-0.

22. Repositories. *GitHub.* [Online] 2018. [Citace: 20. 3 2018.] https://github.com/search?utf8=%E2%9C%93&q=language:JavaScript&type=Repositories&ref=advsearch&l=JavaScript&l.

23. Front-end JavaScript frameworks. *GitHub.* [Online] 2018. [Citace: 20. 3 2018.] https://github.com/collections/front-end-javascript-frameworks.

24. Banks, Alex a Porcello, Eve. *Learning React.* Sebastopol, California : O'Reilly, 2017. ISBN 978-1-491-95462-1.

25. Horyna, Marek. *Single Page Aplikace.* Hradec Králové : Univerzita Hradec Králové, Fakulta informatiky a managementu, 2015. Bakalářská práce.

26. Am, Vipul a Sonpatki, Prathamesh. *ReactJS by Example - Building Modern Web Applications with React.* Birmingham : Packt Publishing, 2016. ISBN 978-1-78528-964-4.

27. Data Binding. *AngularJS.* [Online] 2018. [Citace: 8. 2 2018.] https://docs.angularjs.org/guide/databinding.

28. Deeleman, Pablo. *Learning Angular 2.* Birmingham : Packt Publishing, 2016. ISBN 978-1-785-88207-4.

29. What is NativeScript? *NativeScript.* [Online] 2018. [Citace: 20. 3 2018.] https://www.nativescript.org/about.

30. Morris, Scott. 10 Skills You Need to Land Your First Front End Developer Job. *The Hard Refresh.* [Online] 27. 11 2017. [Citace: 22. 3 2018.] https://skillcrush.com/2017/03/20/front-end-developer-skills/.

31. Kafka, Aleš. *Efektivní nasazování webových aplikací.* Brno : Masarykova Univerzita, Fakulta informatiky, 2015. Diplomová práce.

32. Getting Started. *npm.* [Online] 21. 3 2018. [Citace: 25. 3 2018.] https://docs.npmjs.com/getting-started/installing-node.

33. Jang, Peter. Modern JavaScript Explained For Dinosaurs. *Medium.* [Online] 18. 10 2017. [Citace: 15. 3 2018.] https://medium.com/the-node-js-collection/modern-javascript-explained-for-dinosaurs-f695e9747b70.

34. GULP VS GRUNT VS WEBPACK: COMPARISON OF BUILD TOOLS / TASK RUNNERS. *DA-14.* [Online] 17. 3 2017. [Citace: 17. 3 2018.] https://da-14.com/blog/gulp-vs-grunt-vs-webpack-comparison-build-tools-task-runners.

35. Sample Gruntfile. *GRUNT Documentation.* [Online] 2018. [Citace: 2. 3 2018.] https://gruntjs.com/sample-gruntfile.

36. Javascript (Node.js) with Bitbucket Pipelines. *Bitbucket Support.* [Online] 12. 3 2018. [Citace: 22. 3 2018.] https://confluence.atlassian.com/bitbucket/javascript-node-js-with-bitbucket-pipelines-873891287.html.

37. Nette. *Nette.org.* [Online] 2018. [Citace: 23. 3 2018.] https://nette.org/cs/.

38. Introduction to Spring Framework. *Spring.io.* [Online] 2018. [Citace: 22. 3 2018.] https://docs.spring.io/spring/docs/3.0.x/spring-framework-reference/html/overview.html.

39. Get building. *ASP.NET.* [Online] 2018. [Citace: 22. 3 2018.] https://www.asp.net/.

40. Introducing JSX. *React.* [Online] 2018. [Citace: 22. 3 2018.] https://reactjs.org/docs/introducing-jsx.html.

41. Integrating with Other Libraries. *React.* [Online] 2018. [Citace: 22. 3 2018.] https://reactjs.org/docs/integrating-with-other-libraries.html.

42. react-templates. *GitHub.* [Online] 19. 2 2018. [Citace: 22. 3 2018.] https://github.com/wix/react-templates.

43. Abramov, Dan. @dan_abramov. *Twitter.* [Online] 29. 5 2015. [Citace: 12. 2 2018.] https://twitter.com/dan_abramov/status/604348619102294016.

44. Lisenkov, Anton. How to include JQuery plugins in Angular 2 running via webpack. *GEARHEART web development.* [Online] 3. 10 2016. [Citace: 15. 3 2018.] gearheart.io/blog/how-to-include-jquery-plugins-in-angular-2-running-via-webpack/.

45. Raihan, Shekh Al. Free PSD Corporate Landing Page. *dribbble.* [Online] 3. 12 2015. [Citace: 22. 3 2018.] https://dribbble.com/shots/2385476-Free-PSD-Corporate-Landing-Page.

46. JavaScript. *GetBootstrap.* [Online] 2018. [Citace: 25. 3 2018.] https://getbootstrap.com/docs/4.0/getting-started/javascript/.

47. Grid system. *GetBootstrap.* [Online] 2018. [Citace: 25. 3 2018.] https://getbootstrap.com/docs/4.0/layout/grid/.

48. JavaScript. *Zurb Foundation.* [Online] 2018. [Citace: 4. 4 2018.] https://foundation.zurb.com/sites/docs/javascript.html.

49. XY Grid. *Zurb Foundation.* [Online] 2018. [Citace: 4. 4 2018.] https://foundation.zurb.com/sites/docs/xy-grid.html.

50. Toggler. *Zurb Foundation.* [Online] 2018. [Citace: 4. 4 2018.] https://foundation.zurb.com/sites/docs/toggler.html.

51. facebook/create-react-app. *GitHub.* [Online] 2018. [Citace: 10. 2 2018.] https://github.com/facebook/create-react-app.

52. Ewald, Johannes a Tangelder, Jorik. sass-loader. *webpack.* [Online] 2018. [Citace: 11. 2 2018.] https://webpack.js.org/loaders/sass-loader.

53. QuickStart. *AngularJS.* [Online] 2018. [Citace: 15. 2 2018.] https://angular.io/guide/quickstart.

54. Angular CLI SASS options. *stackoverflow.com.* [Online] 2016. [Citace: 5. 4 2018.] https://stackoverflow.com/questions/36220256/angular-cli-sass-options.

55. react-bootstrap. *GitHub.* [Online] 2018. [Citace: 4. 4 2018.] https://github.com/react-bootstrap/react-bootstrap.

56. twbs/bootstrap. *GitHub.* [Online] 2018. [Citace: 5. 4 2018.] https://github.com/twbs/bootstrap.

57. Components. *React-Bootstrap.* [Online] 2018. [Citace: 5. 4 2018.] https://react-bootstrap.github.io/components/dropdowns/.

58. Dropdowns. *GetBootstrap.* [Online] 2018. [Citace: 25. 3 2018.] https://getbootstrap.com/docs/4.0/components/dropdowns/.

59. Template Syntax. *Vue.js.* [Online] 2018. [Citace: 25. 3 2018.] https://vuejs.org/v2/guide/syntax.html.

60. Collapse. *GetBootstrap.* [Online] 2018. [Citace: 25. 3 2018.] https://getbootstrap.com/docs/4.0/components/collapse/.

7.2 Seznam obrázků

7.3 Seznam tabulek

7.4 Seznam ukázek kódu

8 Přílohy

Popis zprovoznění jednotlivých aplikací, které jsou uloženy na přiloženém DVD.

8.1 Instalace Node.js

https://nodejs.org/en/

Stažení aktuální instalace pro příslušný OS - např. Windows

Zdroj img: https://nodejs.org/en/

Následně se postupuje dle instrukcí instalovaného programu.

Pro kontrolu úspěšného nainstalování node.js na počítači lze v příkazové řádce spustit příkazy

```
node --version
v6.9.2
npm --version
3.10.9
```

8.2 HTML šablony

html-template-bootstrap

Pokud není nainstalován Grunt CLI - https://gruntjs.com/getting-started

parametr -g znamená globální instalaci, začne fungovat v příkazové řádce příkaz: grunt

```
npm install -g grunt-cli
npm install
grunt
```

Šablona se nachází v /www/index.html

html-template-foundation

Pokud není nainstalován Grunt CLI - https://gruntjs.com/getting-started

parametr -g znamená globální instalaci, začne fungovat v příkazové řádce příkaz: grunt

```
npm install -g grunt-cli
npm install
grunt
```

Šablona se nachází v /www/index.html

8.3 Javascriptové aplikace

angular-with-bootstrap

Pokud není nainstalován Angular CLI - https://angular.io/guide/quickstart

parametr -g znamená globální instalaci, začne fungovat v příkazové řádce příkaz: ng

```
npm install -g @angular/cli
npm install
ng serve --open
```

angular-with-foundation

Pokud není nainstalován Angular CLI - https://angular.io/guide/quickstart

parametr -g znamená globální instalaci, začne fungovat v příkazové řádce příkaz: ng

```
npm install -g @angular/cli
npm install
ng serve –open
```

react-with-bootstrap

```
npm install
npm start
```

react-with-foundation

```
npm install
npm start
```

Univerzita Hradec Králové
Fakulta informatiky a managementu
Akademický rok: 2016/2017

Studijní program: Aplikovaná informatika
Forma: Kombinovaná
Obor/komb.: Aplikovaná informatika (ai2-k)

Podklad pro zadání DIPLOMOVÉ práce studenta

PŘEDKLÁDÁ:	ADRESA	OSOBNÍ ČÍSLO
Bc. Vinopal Boštíková Marie	Nám. T.G. Masaryka 617/32, Poděbrady - Poděbrady II	I1600277

TÉMA ČESKY:

Využití frontend frameworků v javascriptových frameworcích a jejich implementace

TÉMA ANGLICKY:

Use frontend frameworks in JavaScript frameworks for their implementation

VEDOUCÍ PRÁCE:

prof. RNDr. PhDr. Antonín Slabý, CSc. - KIKM

ZÁSADY PRO VYPRACOVÁNÍ:

Cíl: Definovat ideální způsob jak implementovat dodanou grafiku do js frameworku (client side rendering). V rámci tématu se přispěje k následujícím problémům: zachování komfortu frontend frameworků (bootstrap, foundation) při vývoji client side rendering web. aplikace. Lze minimalizovat kódování šablon bokem a následně oživovat statiku v aplikaci (problém udržovat 2 frontendy je provozně neudržitelné)? Lze vývoj nastavit tak, aby na něm mohl participovat kodér, který neumí dostatečně javascript?

1. Úvod
2. CSS frontend framework
3. JS frontend framework
4. Trendy a budoucnost JS frontend frameworků
5. Příprava šablony pro JS frontend framework
6. Využití vytvořených šablon v JS frontend frameworku
7. Výsledky srovnání
8. Závěr

SEZNAM DOPORUČENÉ LITERATURY:

Arsenault, C. - Top 10 Front-End Frameworks of 2016, https://www.keycdn.com/blog/front-end-frameworks/
Bhaumik, S. - Bootstrap Essentials, Birmingham, Packt Publishing
Bodmer, M. - Instant Ember.js Application Development How-to, Birmingham, Packt Publishing.
Horton, A., & Vice, R. - Mastering React. Birmingham, Packt Publishing
Murray, N., Coury, F., Lerner, A., & Taborda, C. - ng-book 2, San Francisco, Fullstack.io

Podpis studenta: Vinopal Boštíková **Datum:** 11.10.2017

Podpis vedoucího práce: **Datum:** 11.10.2017