Gymnázium, Praha 7, Nad Štolou 1

Maturitní práce

Tvorba dynamického webu

Autor práce: David Straka

Vedoucí práce: Mgr. Jiří Pavlíček

O8A

2017 / 2018

Abstrakt

Cílem této maturitní práce bylo vytvořit dynamický web pro prezentaci fiktivní cestovní agentury. Takový web by měl být responzivní a optimalizovaný.

V teoretické části práce je popsáno, jakým způsobem bude problém řešen. Praktická část pak obsahuje ukázky kódu a funkčnosti webu. V závěru je zhodnoceno, jak dobře byl cíl splněn a co by šlo udělat lépe.

Abstract

The objective of this work was to create a dynamic website for the presentation of a fictional travel agency. Such website shall be responsive and optimised.

In the theoretical part, there is described how will the problem be solved. The practical part contains demonstrations of the source code and the website functionality. In the ending, there is an evaluation of how well was the objective accomplished and what could have been done better.

Prohlášení

Prohlašuji, že jsem maturitní práci vypracoval samostatně, použil jsem pouze podklady uvedené v přiloženém seznamu a postup při zpracování a dalším nakládání s prací je v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů.

V Praze dne …….. ……

Podpis autora práce

Poděkování

Tímto bych rád poděkoval vedoucímu mé maturitní práce, panu Mgr. Jiřímu Pavlíčkovi, za jeho vstřícnost, trpělivost, ochotu a čas, které pro mě byly nepostradatelné pro řádné zhotovení mé práce.

Obsah

Seznam obrázků

Úvod

Ve své maturitní práci se budu zabývat tvorbou dynamického webu pro prezentaci fiktivní cestovní agentury. Výsledné webové stránky budou využívat moderní webové technologie a budou responzivní, aby je bylo možno pohodlně používat jak na mnoha různých zařízeních – od telefonů a tabletů, přes laptopy a stolní počítače, až po televize a mnoho dalších. Také budou optimalizované pro rychlé načítání a zmenšení datových přenosů. Prezentace by měla návštěvníkům webu poskytnout informace o oné firmě a o produktech, jež nabízí. Nabídne uživateli i nějakou formu interakce.

V teoretické části mé práce nejprve popíšu, jakým způsobem budu problém řešit. V praktické části zahrnu stěžejní části kódu a ukázky funkčnosti hotových webových stránek. V závěru se pak zamyslím nad tím, jaká část problému byla vyřešena, co ještě zbývá vyřešit a jaké jsou další možné zlepšení webových stránek.

Téma mé maturitní práce „Tvorba dynamického webu“ jsem si vybral, protože se webovým technologiím věnuji již delší dobu a tvorbu webových stránek bych označil svým koníčkem. Tato činnost mě baví, protože v ní mohu skloubit kreativitu a nadšení k moderním technologiím. Webové standardy se neustále vyvíjí a neustále se tak nabízí nové věci, kterým se mohu naučit. Rád bych se tomu věnoval i nadále, ať už při studiu na vysoké škole, nebo v profesním životě.

1. Teoretická část

„*Website (nebo web site, běžně též web, řidčeji webové místo, anglicky doslova „místo v pavučině“) označuje kolekci webových stránek, obrázků, videí a ostatních souborů, které jsou uloženy na jednom nebo více webových serverech a jsou dostupné pomocí Internetu*.“ „*Pro tyto soubory je společné to, že (přestože mohou být fyzicky na více místech) tvoří jeden logický celek – co do obsahu, účelu, majitele, administrace, obchodního modelu (u komerčních websites) a dalších*.“ Všechny weby dohromady spojené prostřednictvím Internetu a hypertextových odkazů pak tvoří World Wide Web. [1]

Pro vytvoření webové stránky je potřeba mnoha různých součástí. Mezi ty základní patří HTML, kaskádové styly (CSS), JavaScript a multimédia.

1.1. HTML

HTML je značkovací jazyk, který umožňuje tvorbu strukturovaných dokumentů, jež je dále možno dále obohatit o multimédia (obrázky, zvuk a video) a vzájemně propojit hypertextovými odkazy. HTML dokument zpravidla obsahuje deklaraci typu dokumentu a kořenový prvek dělený na hlavičku a tělo.

Typická hlavička dokumentu obsahuje informace o kódování souboru, o jazyku nebo autorovi obsahu, titulek, krátký popis a klíčová slova obsahu, kaskádové styly vložené přímo nebo externě, skripty vložené přímo nebo externě, ikonky webu a další.

Tělo dokumentu pak obsahuje samotný obsah zobrazený uživateli. Tím jsou zpravidla nadpisy, odstavce, seznamy, tabulky, formuláře, odkazy, obrázky, videa, zvuky a mnoho dalších. V těle dokumentu mohou být také jako v hlavičce obsaženy styly a skripty. [2]

1.2. Kaskádové styly

Kaskádové styly jsou počítačovým jazykem, jenž určuje způsob zobrazení dokumentu psaného ve strukturovaném jazyce – tedy mimo jiné v jazyce HTML.

Soubor kaskádových stylů je tvořen pravidly, která se skládají z jednoho nebo více selektorů a deklaračního bloku. Selektory udávají, ke kterým prvkům strukturovaného dokumentu se bude následující deklarační blok vztahovat. V deklaračním bloku nalezneme deklarace – dvojice názvů vlastností a jejich hodnot. Deklarace tak mohou udávat například barvu písma nebo pozadí, velikost a styl písma, vzhled rámečku, velikost okrajů a další. [3]

1.2.1. Preprocesory

„*Preprocesor je počítačový program, který zpracovává vstupní data tak, aby výstup mohl dále zpracovávat jiný program. Preprocesor je často používán pro předzpracování zdrojového kódu před vlastní kompilací*.“ [4]

Preprocesory využiji při psaní kaskádových stylů. Ty totiž není nutno psát přímo, ale můžeme využít jazykových nadstaveb jako například SCSS, SASS nebo LESS. Tyto nadstavby nám umožnují používat pro psaní kaskádových stylů odlišnou syntaxi a rozšířenou funkčnost.

V případě SCSS, který budu používat já (tedy konkrétně ve formě Node-sass – modulu pro Node.js, který zpracovává SCSS), to jsou například proměnné hodnot, proměnné skupiny deklarací, vnořování deklaračních bloků, matematické výpočty, podmínky nebo importování stylů z dalších souborů.

Některé z těchto funkcí se pomalu začínají objevovat přímo v kaskádových stylech. Preprocesory však mají výhodu, že nezávisí na adaptaci funkčnosti prohlížeči a třeba v případě výpočtů není mnohdy důvod, proč je neprovést ještě před nasazením kaskádových stylů do produkční verze webu.

Dalším preprocesorem, který využiji, je PostCSS s pluginem Autoprefixer. PostCSS je nástroj, který vygeneruje z kaskádových stylů abstraktní strom, se kterým pak mohou různě manipulovat další pluginy. Z upraveného stromu pak PostCSS zpětně vygeneruje kaskádové styly. [5]

Autoprefixer je plugin pro PostCSS, jenž přidá k názvům vlastností kaskádových stylů předpony prohlížečů, je-li tomu pro cílenou skupinu prohlížečů, které chceme podporovat, nutné. Tyto předpony jsou nutné u nových vlastností, které ještě nemají univerzální podporu a je tak časově obtížné neustále kontrolovat, u kterých vlastností je předpon ještě potřeba.

1.2.2. Knihovny

„*Knihovna (anglicky library) je v informatice označení pro souhrn procedur a funkcí, často také konstant a datových typů (v objektovém programování též tříd, objektů a zdrojů), který může být využíván více počítačovými programy. Knihovny usnadňují programátorovi tvorbu aplikací tím, že umožňují využití hotového kódu, použití jednou vytvořeného kódu v jiných programech; při týmové práci mohou sloužit k dělbě práce*.“ [6]

Většina prohlížečů se alespoň trochu liší ve výchozím nastavení hodnot některých vlastností. Ve své práci proto využiji knihovnu kaskádových stylů Normalize.css, která se snaží o

sjednocení chování prohlížečů, co se týče způsobu zobrazování dokumentu. Pro její importování využiji PostCSS plugin postcss-import.

1.3. JavaScript

JavaScript je vysokoúrovňový objektově orientovaný interpretovaný programovací jazyk, který se na webových stránkách využívá nejčastěji k obohacení stránky o interaktivní prvky, kdy kód skriptu (v samostatném souboru nebo přímo vložený do HTML dokumentu) si stáhne prohlížeč uživatele a skript běží na jeho straně. Využití JavaScriptu jsou však rozsáhlá a dnes se již běžně využívá i na straně serveru. [7]

1.3.1. Transpilery

„*Transpiler je typ překladače, který přeloží zdrojový kód z jednoho programovacího jazyka do jiného. Transpiler pracuje s jazyky na přibližně stejné úrovni abstrakce, zatímco tradiční kompilátor kompiluje jazyk na vysoké úrovni abstrakce do jazyka na nízké úrovni abstrakce*.“ [8]

Při své práci využiji transpileru Babel, který mi umožní užívat při psaní JavaScriptu jeho nejnovější syntaxi, která ještě není podporována aktuálními verzemi webových prohlížečů. Z této syntaxe Babel vygeneruje ekvivalentní kód v syntaxi starší, se kterou už si současné prohlížeče poradí.

1.4. Obrázky

Mezi zjevně nejpoužívanější typ multimédií na webu patří obrázky. Ty se mohou vyskytovat v rastrové (bitmapové), nebo vektorové podobě.

Rastrové obrázky uchovávají data v podobě pixelů uspořádaných do mřížky, kdy každý pixel udává, jaká barva (případně s jakou transparentností) má být v daném místě zobrazena. Mnoho pixelů (dnes řádově statisíce až miliony) dohromady pak tvoří celý obraz. Tento typ obrázků je využíván, je-li potřeba mnoha detailů a různých barev – typicky třeba u fotografií. Rastrovým formátem je například JPEG, PNG nebo GIF.

Vektorové obrázky se skládají z geometrických útvarů, jako jsou body, přímky a úsečky, křivky, polygony. Tyto útvary jsou pak uloženy v podobě matematických vyjádření. Jejich výhodou oproti rastrovým obrázkům je možnost změny velikosti bez ztráty kvality. Jednoduché obrázky navíc zaberou výrazně méně místa úložiště. Nejčastěji je najdeme v podobě různých log a ikonek. Vektorovým formátem je pak třeba SVG nebo EPS. [9]

1.4.1. Optimalizace obrázků

Protože obrázky tvoří značnou část datového toku webu, je patřičné se snažit tento tok co nejvíce možno zredukovat.

Pro fotografie tak využiji ztrátového rastrového formátu JPEG. Rozlišení obrázků s pomocí Node.js modulu Jimp zmenším na vhodnou velikost (okolo 2 Mpx). Následně využiji JPEG kodéru Guetzli (tedy konkrétně Node.js modulu imagemin-guetzli) pro pokročilou kompresy. Guetzli provede sérii návrhů komprese obrázku, z nichž pomocí algoritmu Butteraugli vybere ten s nejmenší psychovizuální chybou. [10] Dostaneme tak obrázek s vynikajícím poměrem velikosti a člověkem vnímané kvality. Nevýhodou je jeho pomalost a paměťová náročnost – komprese zabere přibližně minutu času CPU a 300 MB RAM na jeden Mpx obrázku. [11]

Přestože jednoduché vektorové obrázky nezabírají mnoho místa, je možno jejich velikost také výrazně zredukovat. V případě loga webu, pro jehož tvorbu využiji program Inkscape, použiji již v Inkscape obsaženou možnost optimalizace SVG. Kód SVG loga pak celý vložím přímo do HTML. Tím docílím eliminace jedné HTTP žádosti klienta na server, což je dobré pro rychlost načítání stránky.

Pro ikonky pak využiji open source set ikonek Material icons od Googlu, kde je s každou ikonkou obsažena i již optimalizovaná verze. Abych opět snížil počet HTTP žádost prohlížeče uživatele na server webu, vytvořím navíc z ikonek takzvaný sprite sheet, kdy jsou všechny ikonky obsaženy v jednom SVG souboru. Toho docílím pomocí Node.js modulu svgstore.

1.5. Minifikace

…

2. Praktická část

2.1. Ukázky zdrojového kódu

2.1.1. Progresivní lazy loading obrázků

```
const imgLoaded = (img, fullImg, callback) => {
    img.classList.add('loaded');
    img.setAttribute('src', fullImg.getAttribute('src'));
    img.removeAttribute('data-src');
    if (callback)
        callback(img);
};

const loadImg = (img, callback) => {
    let fullImg = document.createElement('img');
    fullImg.setAttribute('src', img.getAttribute('data-src'));
    fullImg.addEventListener('load', imgLoaded
        .bind(null, img, fullImg, callback));
    let cvs = document.createElement('canvas');
    let ctx = cvs.getContext('2d');
    const size = [img.naturalWidth, img.naturalHeight];
    [cvs.width, cvs.height] = size;
    ctx.filter = 'blur(2px)';
    ctx.drawImage(img, 0, 0, size[0], size[1]);
    if (!img.classList.contains('loaded'))
        img.setAttribute('src', cvs.toDataURL());
};

const loadImgs = (imgs, callback) => imgs
    .forEach(img => loadImg(img, callback));

export default loadImgs;
```

Pro mou první ukázku zdrojového kódu jsem si vybral funkce, které zajišťují progresivní lazy loading obrázků. Lazy loading je proces, kdy obrázky načteme až po načtení ostatního, důležitějšího obsahu dokumentu. Je-li tento proces progresivní, znamená to, že je uživateli nejprve zobrazena nějaká částečná náhrada načítaného obrázku.

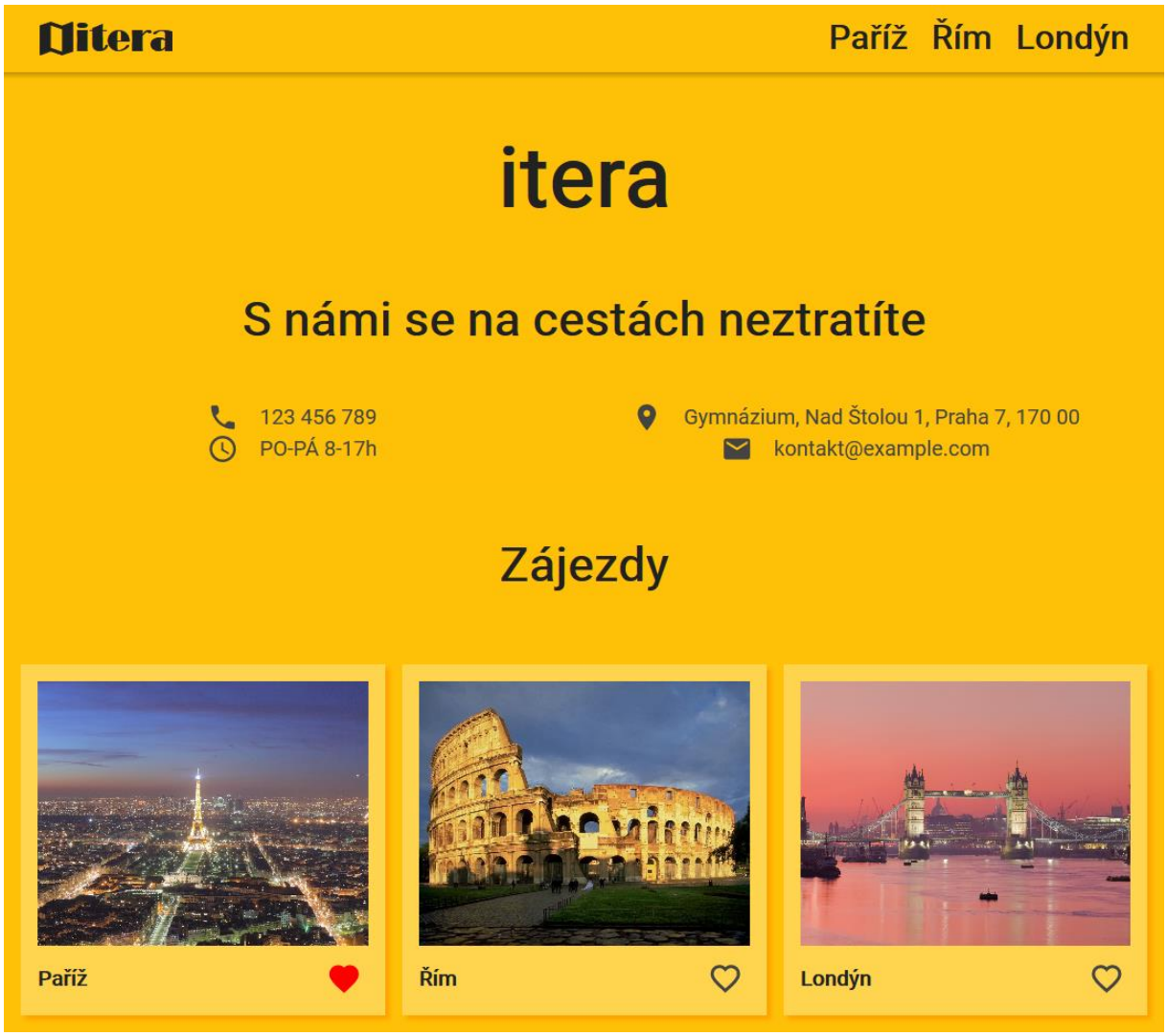
Fotografie na stránce jsem tak nahradil miniaturními náhledy, každý o velikosti cca 1 kB. Tyto náhledy jsou navíc vloženy přímo do HTML dokumentu ve formě data URI, aby se snížil počet HTML žádostí klienta na server.

Po načtení důležitého obsahu dokumentu je díky události DOMContentLoaded zavolána funkce, která najde na stránce všechny obrázky s třídou „lazy“. Tyto obrázky jsou pak podány v podobě proměnné „imgs“ funkci loadImgs, která pro každý z nich zavolá funkci loadImg. Navíc můžeme funkci podat i callback (funkci, která se spustí po načtení obrázku). Díky tomu mohu obrázkům umožnit funkci další funkci přiblížení až poté, co se plně načtou.

Funkce loadImg začne načítat plnou verzi obrázku a zatím, co se obrázek načítá, vytvoří rozostřenou verzi náhledové verze obrázku a nahradí jí původní náhled. Rozostření pomáhá zakrýt ošklivé bloky v náhledu způsobené těžkou kompresí a zmenšením a následným zvětšením na cílenou velikost na stránce. Po načtení plné verze obrázku je jí náhled na stránce nahrazen funkcí „imgLoaded“ a je-li nějaká callback funkce (například ta pro aktivaci přiblížení), je zavolána.

2.2. Ukázky funkčnosti webu

2.2.1. Hlavní stránka



Obrázek 1: Hlavní stránka

Na prvním obrázku vidíme hlavní stránku webu.

V horní části se nachází menu. V jeho levé části nalezneme logo firmy s odkazem na hlavní stránku. V pravé části se pak nachází odkazy na produkty firmy. Tato horní lišta se posouvá při posouvání stránky (tedy zůstává vždy nahoře) a na každé stránce firmy stejná.

Pod horní lištou vidíme název a slogan firmy následovaný kontaktními údaji s vektorovými ikonkami. Ve spodní části stránky je seznam produktů (v tomto případě tedy zájezdů) firmy. Karta produktu se skládá vždy z úvodní fotografie produktu, názvu produktu a tlačítka pro přidání produktu do oblíbených. Již oblíbené produkty pak mají barevnou indikaci. Fotografie i název slouží zároveň také jako odkaz. Počet produktů i velikost jejich karet se přizpůsobuje velikosti okna prohlížeče.

2.2.2. Stránka produktu

Londýn

Londýn je hlavní město Spojeného království Velké Británie a Severního Irska ležící na jihovýchodě země při ústí řeky Temže. Londýnská City je jedno z největších světových obchodních center.

V tomto **dvoudenním** výletě si prohlédneme mimo jiné například následující:

Tower Bridge

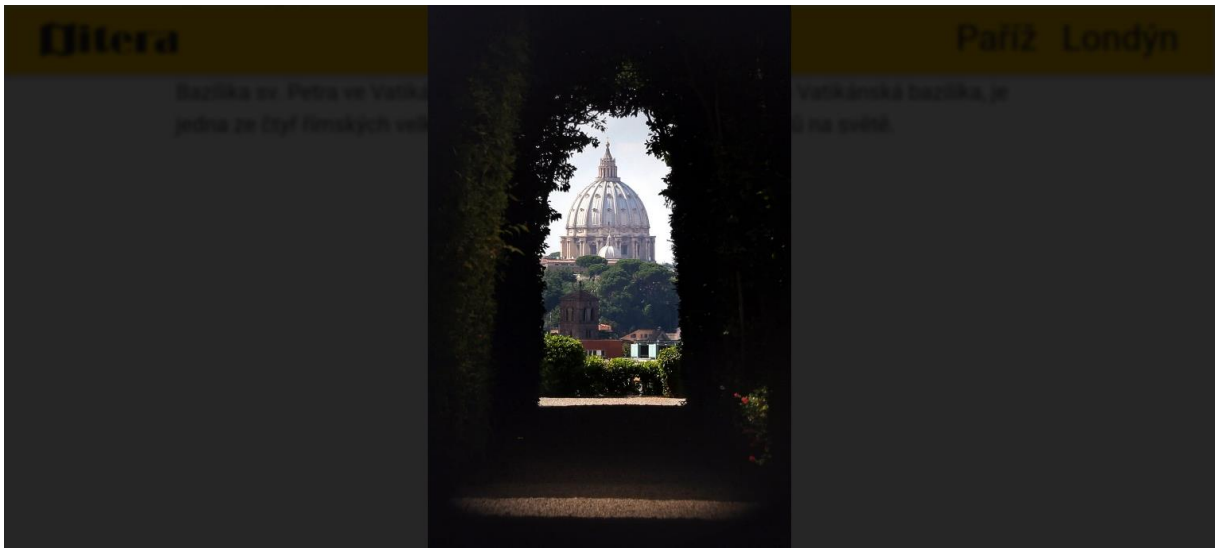
Tower Bridge je zvedací most v Londýně nad řekou Temží. Vedle něj stojí Tower, podle níž se jmenuje.

Obrázek 2: Stránka produktu

Na druhé ukázce můžeme vidět stránku produktu (v tomto případě zájezdu do Londýna).

V horní části se opět nachází lišta s logem a menu. Pod ní najdeme úvodní obrázek produktu, cenu a tlačítko pro přidání do oblíbených. V hlavní části stránky pak uživatel najde

všechny potřebné informace k produktu a další fotografie. Fotografie je možno stisknutím levého tlačítka myši (popřípadě klepnutím prstu na dotykové obrazovce) přiblížit.



Obrázek 3: Přiblížení obrázku

Při přiblížení fotografie vyplní dostupný prostor okna prohlížeče při zachování jejího poměru stran. Je vertikálně i horizontálně vycentrována (díky vyplnění okna uživatel vidí nanejvýš jednu úroveň centrování). Pozadí je tmavé a z části průhledné, uživatel tak ví, že se stále nachází na stránce, ale není rušen obsahem. Po dobu přiblížení je deaktivováno posouvání stránky. Pro zrušení stačí kdekoli v prostoru stránky opět stisknout levé tlačítko myši.

Závěr

V mé maturitní práci jsem zpracoval téma „Tvorba dynamického webu“.

V rámci teoretické části jsem se zamyslel nad tím, jak se bude daný problém řešit. To zahrnovalo využití zavedených webových standardů, moderních nástrojů rozšiřujících práci na nich nebo způsoby optimalizace webu i práce na něm.

V praktické části jsem vypracoval webové stránky pro prezentaci fiktivní cestovní agentury. Tyto stránky využívají moderní webové technologie i zavedené standardy. Jsou responzivní a poskytují tedy dobrou uživatelskou zkušenost jak na stolních počítačích a laptopech, tak na mobilních zařízeních. Jsou optimalizované a nespotřebovávají tedy zbytečné množství dat ani serveru, ani koncovému uživateli. Navíc se díky tomu rychleji načítají. Stránky poskytují i interaktivní prvek v podobě přibližování obrázků nebo ukládání produktů mezi oblíbené (nezávisle na oblíbených stránkách v prohlížeči). Web běží bez větších problémů na větší části současných webových prohlížečů.

Zadání maturitní práce tak bylo dle mého názoru zcela splněno a byly úspěšně vytvořeny i skutečnou firmou použitelné webové stránky. Jako možná vylepšení bych přidal možnost objednání si produktu nebo nějakou formu komunikace s firmou přímo z webu (třeba ve formě formuláře, jenž odešle email, nebo ve formě chatu se zaměstnancem).

Během práce na své maturitní práci jsem si vyzkoušel tvorbu firemních webových stránek s využitím moderních technologií. Práce mě bavila, řekl bych, že byla úspěšná a věřím, že si z ní do budoucna mnoho odnesu.

Seznam pramenů, literatury a internetových zdrojů

[1] Wikipedie: Otevřená encyklopedie: Website [online]. c2017 [citováno 11. 03. 2018]. Dostupný z WWW: https://cs.wikipedia.org/w/index.php?title=Website&oldid=15407058

[2] Wikipedie: Otevřená encyklopedie: HyperText Markup Language [online]. c2018 [citováno 11. 03. 2018]. Dostupný z WWW: https://cs.wikipedia.org/w/index.php?title=HyperText_Markup_Language&oldid=15892640

[3] Wikipedia, The Free Encyclopedia: Cascading Style Sheets [online]. c2018 [citováno 11. 03. 2018]. Dostupný z WWW: https://en.wikipedia.org/w/index.php?title=Cascading_Style_Sheets&oldid=826758636

[4] Wikipedie: Otevřená encyklopedie: Preprocesor [online]. c2015 [citováno 14. 03. 2018]. Dostupný z WWW: https://cs.wikipedia.org/w/index.php?title=Preprocesor&oldid=12591061

[5] Wikipedia, The Free Encyclopedia: PostCSS [online]. c2018 [citováno 14. 03. 2018]. Dostupný z WWW: https://en.wikipedia.org/w/index.php?title=PostCSS&oldid=819315397

[6] Wikipedie: Otevřená encyklopedie: Knihovna (programování) [online]. c2017 [citováno 15. 03. 2018]. Dostupný z WWW: https://cs.wikipedia.org/w/index.php?title=Knihovna_(programov%C3%A1n%C3%AD)&oldid=15611496

[7] Wikipedia, The Free Encyclopedia: JavaScript [online]. c2018 [citováno 19. 03. 2018]. Dostupný z WWW: https://en.wikipedia.org/w/index.php?title=JavaScript&oldid=831151277

[8] Wikipedie: Otevřená encyklopedie: Transpiler [online]. c2017 [citováno 18. 03. 2018]. Dostupný z WWW: https://cs.wikipedia.org/w/index.php?title=Transpiler&oldid=14747196

[9] Wikipedie: Otevřená encyklopedie: Vektorová grafika [online]. c2018 [citováno 16. 03. 2018]. Dostupný z WWW: https://cs.wikipedia.org/w/index.php?title=Vektorov%C3%A1_grafika&oldid=15746310

[10] Google Developers: Web Fundamentals: Automating image optimization [online]. c2018 [citováno 17. 03. 2018]. Dostupný z WWW: https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/automating-image-optimization/

[11] GitHub: Guetzli: REDME [online]. c2017 [citováno 17. 03. 2018]. Dostupný z WWW: https://github.com/google/guetzli/blob/master/README.md