

Gymnázium, Praha 7, Nad Štolou 1

Maturitní práce  
**Tvorba dynamického webu**

Autor práce: David Straka

Vedoucí práce: Mgr. Jiří Pavlíček

Třída: O8A

Školní rok: 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 optimized.

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 příloženém 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

Úvod.....	7
1 Teoretická část.....	8
1.1 HTML.....	8
1.2 Kaskádové styly.....	8
1.2.1 Preprocesory .....	9
1.2.2 Knihovny .....	9
1.3 JavaScript .....	10
1.3.1 Transpilery .....	10
1.4 Obrázky .....	10
1.4.1 Optimalizace obrázků .....	11
1.5 Minifikace.....	11
1.6 Verzování.....	12
1.7 Node.js.....	12
2 Praktická část.....	14
2.1 Ukázky zdrojového kódu.....	14
2.1.1 Progresivní lazy loading obrázků .....	14
2.1.2 Zobrazování odkazů v menu na telefonu .....	15
2.2 Ukázky funkčnosti webu .....	17
2.2.1 Hlavní stránka .....	17
2.2.2 Stránka produktu.....	19
Závěr.....	22
Seznam pramenů, literatury a internetových zdrojů.....	23

## Seznam obrázků

Obrázek 1: Hlavní stránka.....	17
Obrázek 2: Hlavní stránka zobrazená na telefonu – část 1 .....	18
Obrázek 3: Hlavní stránka zobrazená na telefonu – část 2 .....	18
Obrázek 4: Menu zobrazené na telefonu.....	18
Obrázek 5: Stránka produktu.....	19
Obrázek 6: Přiblížení obrázku.....	20
Obrázek 7: Stránka produktu zobrazená na telefonu – část 1 .....	21
Obrázek 8: Stránka produktu zobrazená na telefonu – část 2 .....	21

# 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 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 jaká jsou další možná 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 jako svůj koníček. Tato činnost mě baví, protože v ní mohu skloubit kreativitu s nadšením k moderním technologiím. Webové standardy se neustále rozvíjí a nepřestávají tak nabízet 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 (... , běžně též web, ...) 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 ...“* 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 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 využijí při psaní kaskádových stylů. Ty totiž není nutno psát přímo, ale můžeme využít jazykových nadstavb jako například SCSS, SASS nebo LESS. Tyto nadstavby nám umožňují 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 využijí, 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 ... je v informatice označení pro souhrn procedur a funkcí, často také konstant a datových typů ..., 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 využijí knihovnu kaskádových stylů Normalize.css, která se snaží o sjednocení chování prohlížečů, co se týče způsobu zobrazování dokumentu. Pro její importování využijí 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 zabírají výrazně méně místa úložiště. Nejčastěji je najdeme v podobě různých log a ikon. 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 využijí ztrátového rastrového formátu JPEG. Rozlišení obrázků s pomocí Node.js modulu `Im` zmenším na vhodnou velikost (okolo 2 Mpx). Následně využijí JPEG kodéru `Guetzli` (tedy konkrétně Node.js modulu `imagemin-guetzli`) pro pokročilou kompresi. `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 využijí program `Inkscape`, použijí 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 využijí open source set ikon `Material icons` od Googlu, kde je s každou ikonkou obsažena i již optimalizovaná verze. Abych opět snížil počet HTTP žádostí prohlížeče uživatele na server webu, vytvořím navíc z ikon `Material icons` takzvaný `sprite sheet`, kdy jsou všechny ikonky obsaženy v jednom SVG souboru. Toho docílím pomocí Node.js modulu `svgstore`.

## 1.5 Minifikace

Minifikace je proces, při němž se z kódu odstraní komentáře a zbytečné mezery. Jde-li nějaká část kódu napsat kratší formou (například zkrácení jména barvy nebo použití obecnějšího jména vlastnosti v kaskádových stylech), je jí též nahrazena.

Minifikační nástroje navíc většinou provádějí takzvaný `mangling` jmen. To je proces, kdy se jména proměnných, funkcí, konstant, zkrátka všeho co jde při psaní kódu nějak pojmenovat, nahradí nejkratším možným jménem (v praxi si tedy i rozsáhlé programy vystačí se dvěma znaky na jméno). [12]

Samotné odstranění mezer by se dalo označit za zbytečné kvůli kompresi dat prováděné při správném nastavení automaticky serverem. Většinu ostatních procedur minifikace ale komprese nenahradí. Nejlepších výsledků redukce přenesených dat tak dosáhneme použitím obou.

K minifikaci použiju Node.js moduly `html-minifier` (pro HTML) a `uglify-js` (pro JavaScript). Minifikaci kaskádových stylů pak provede již preprocesor SCSS.

## 1.6 Verzování

„Verzování je uchovávání historie veškerých změn provedených v informacích nebo datech. Nejčastěji se používá pro sledování změn ve zdrojových kódech softwaru během jeho vývoje, ale verzovat lze všechny počítačové soubory, a v případě potřeby se verzují i primární data.“ [13]

Tento proces je důležitý, protože šetří čas při hledání nějaké chyby, která v předchozí verzi kódu nebyla. Pro verzování využiji GitHub, což je služba postavená na open source verzovacím nástroji Git.

## 1.7 Node.js

Node.js je prostředí postavené na JavaScriptovém jádru V8 z open source prohlížeče Chromium. Toto prostředí umožňuje používat JavaScriptové programy na počítači nebo serveru.

Knihovny pro Node.js se většinou označují jako moduly nebo balíky. Tyto moduly je možno jednoduše stáhnout z příkazového řádku z ekosystému npm, což je největší ekosystém open source knihoven na světě. [14]

Node.js mi umožní spustit JavaScriptový kód, kterým při tvorbě webu budu provádět většinu v předchozích podkapitolách popsanych úkonů, jako jsou minifikace, preprocessing, optimalizace obrázků nebo tvorbu sprite sheets.

Dále ho využiji společně s modulem `rollup` pro takzvaný bundling JavaScriptového kódu. Ten totiž budu psát ve formě mnoha malých modulů pro lepší udržitelnost kódu. Pro lepší výkon webu (díky menšímu počtu HTTP žádostí na server) je pak ale sloučím dohromady v jeden skript. Právě tomuto procesu se říká bundling.

Při tvorbě produkčního kódu (to je optimalizovaného kódu) využiji moduly `fs-extra` a `globby`. `Fs-extra` rozvíjí nativní modul `fs` pro práci se souborovým systémem. `Globby` umožňuje použití intuitivních vzorů pro selekci souborů.

Pro založení lokálního serveru pro testování stránek zase využiji modul `http-server`. Modul `datauri` mi umožní přímé vkládání malých rastrových obrázků do HTML ve formě datového schématu `data URI` a tím tak dosáhnout dalšího snížení počtu HTTP žádostí. Pro automatické

spuštění některých skriptů při upravení specifikovaných souborů použiju modul onchange. Modul npm-run-all mi pak pomocí jednoduchého API umožní spouštět některé z těchto skriptů paralelně a tím dosáhnout rychlostního zlepšení.

## 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));
```

Pro mou první ukázkou 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 mohou 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.

### 2.1.2 Zobrazování odkazů v menu na telefonu

```
let menuOn = false;

const menuToggle = () => {
  if (menuOn) {
    document.body.classList.remove('modal-small-on');
    doc.menu.classList.remove('on');
    doc.modal.removeEventListener('click', menuToggle);
    menuOn = false;
  } else {
    document.body.classList.add('modal-small-on');
    doc.menu.classList.add('on');
    doc.modal.addEventListener('click', menuToggle);
    menuOn = true;
  }
};

const prepDoc = () => {
  doc.menuToggle.addEventListener('click', menuToggle);
};
```

Pro druhou ukázkou jsem zvolil jednoduché funkce, jež umožňují uživateli na zařízeních s malou obrazovkou zobrazit seznam odkazů v menu (který je ve výchozím stavu skryt).

Po načtení skriptu je zavolána funkce „prepDoc“, která tlačítku pro zobrazení menu přiřadí funkci „menuToggle“ pro událost click. Událost click je aktivována kliknutím myši nebo dotykem na dotykové obrazovce.

Při zmáčknutí aktivačního tlačítka je tedy funkce „menuToggle“ zavolána a ta pak kontroluje, zdali je proměnná „menuOn“ pravdivá, nebo ne (ve výchozím stavu pravdivá není).

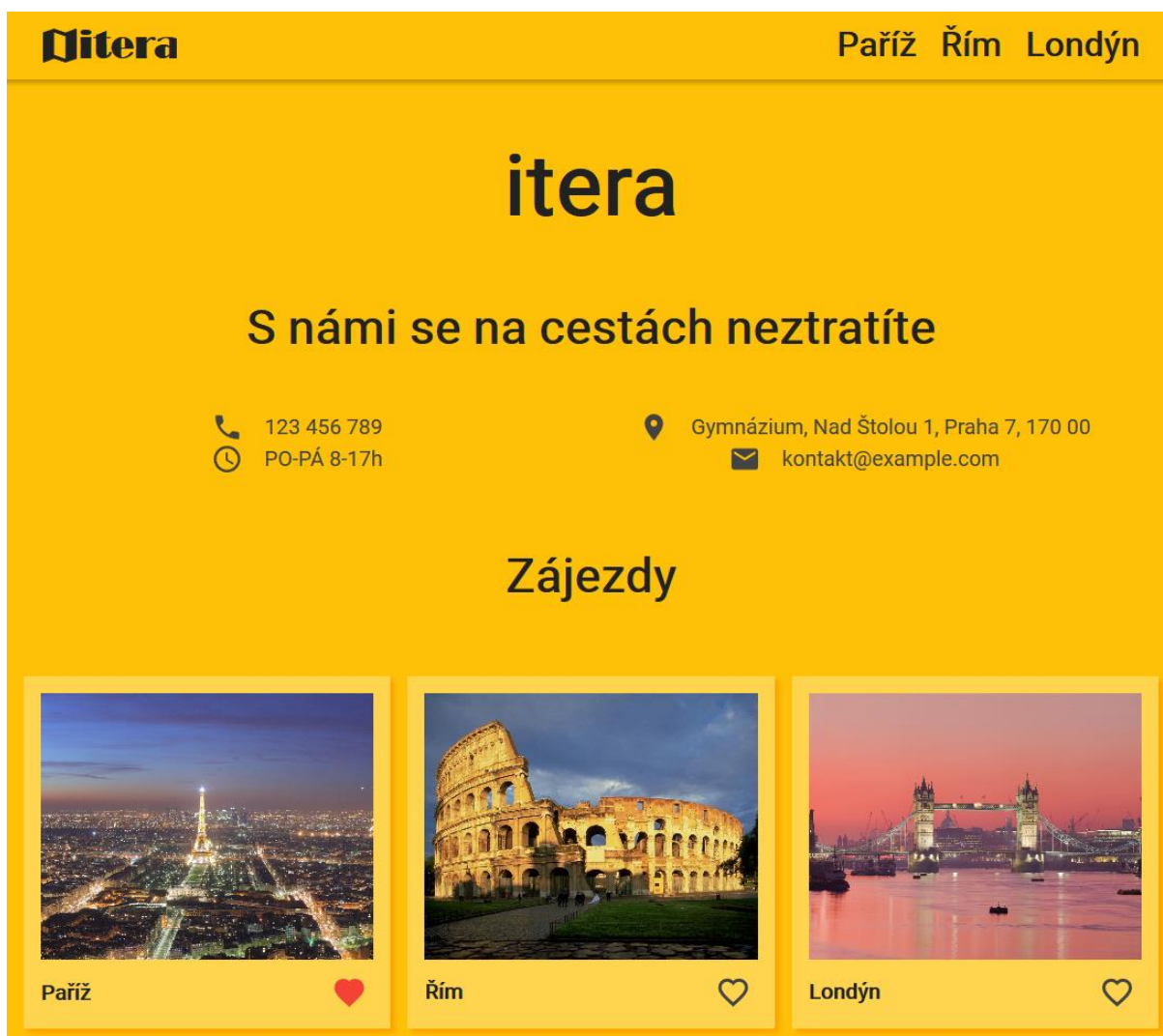
Pokud pravdivá je, je z těla stránky a z prvku s třídou menu odstraněná třída, která indikuje zobrazení odkazů. Z černého částečně průhledného podkladu je navíc odstraněn posluchač události click, protože je tento podklad využíván i při přiblížení obrázku, kde je žádoucí jiné chování. Seznam s odkazy i podklad jsou pak díky kaskádovým stylům, které se řídí odstraněnými třídami, skryty. Nakonec je nastavena proměnná „menuOn“ na nepravdivou hodnotu.

Pokud pravdivá není, je proces inverzní. Třídy jsou tedy přidány, stejně tak posluchač události a nakonec je proměnná „menuOn“ nastavena na pravdivou.



## 2.2 Ukázky funkčnosti webu

### 2.2.1 Hlavní stránka



Obrázek 1: Hlavní stránka

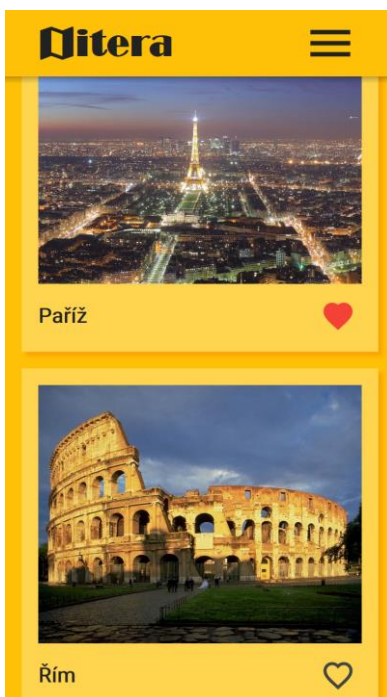
Na prvním obrázku vidíme hlavní stránku webu.

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.

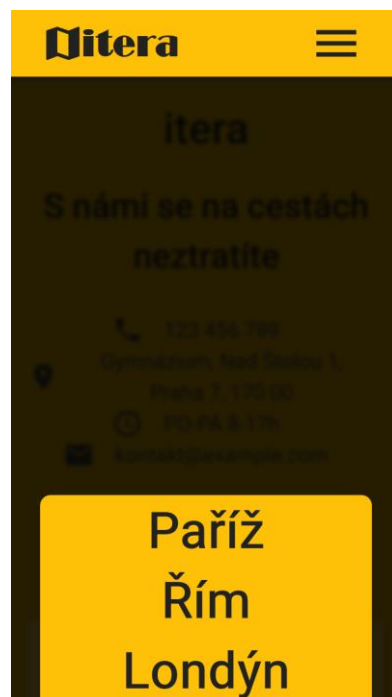
Pod horní lištou vidíme název a slogan firmy následovaný kontaktními údaji s vektorovými ikonkami. Ve spodní části stránky je seznam produktů (v tomto případě tedy zájezdů) firmy. Karta produktu se skládá vždy z úvodní fotografie produktu, názvu produktu a tlačítka pro přidání produktu do oblíbených. 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.



Obrázek 2: Hlavní stránka zobrazená na telefonu – část 1



Obrázek 3: Hlavní stránka zobrazená na telefonu – část 2



Obrázek 4: Menu zobrazené na telefonu

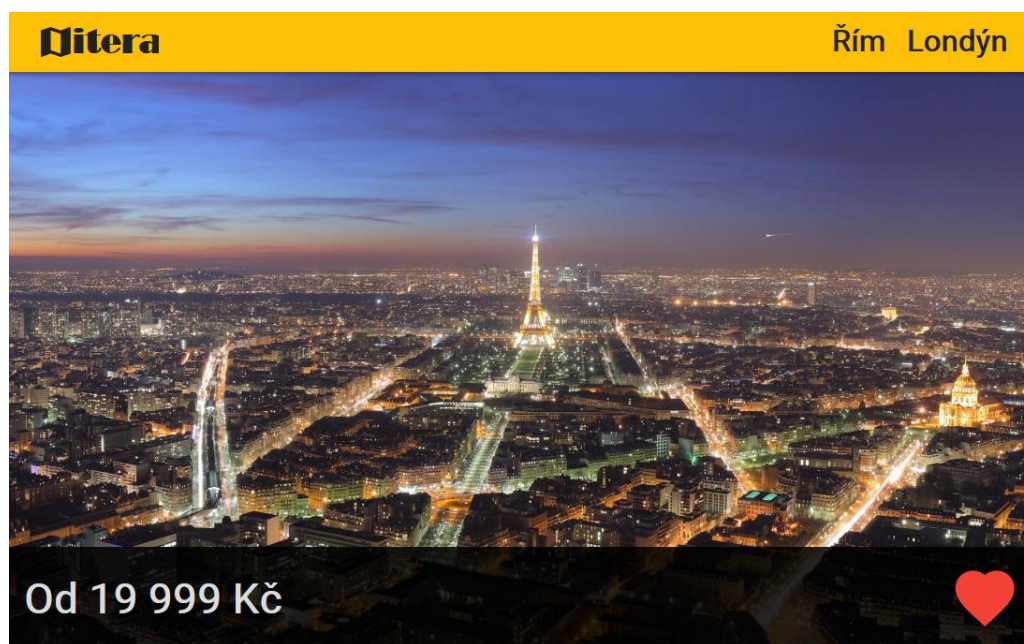
Obrázky 2 a 3 zobrazují hlavní stránku na mobilním telefonu.

Pravá část menu s odkazy na produkty je skryta pro ušetření prostoru obrazovky. Uživatel ji může zobrazit kliknutím na ikonku se třemi horizontálními pruhy. Tato ikonka se na webu k tomuto účelu často používá a měla by tak být intuitivní pro většinu návštěvníků stránky.

Seznam s kontaktními údaji i seznam produktů jsou pak zmenšeny do jediného sloupce.

Menu s odkazy se nachází na obrázku 4. Odkazy se nachází u spodního kraje obrazovky pro lepší ovladatelnost jednou rukou na mobilních zařízeních s většími obrazovkami. Seznam odkazů je podložen černým poloprůhledným podkladem a je vypnuto posouvání stránky.

## 2.2.2 Stránka produktu



### Paříž

Paříž je hlavní a zároveň největší město Francie. Toto město má dlouhou historickou tradici a tak zde najdeme nespočet památek.

### Eiffelova věž

Věž byla předlohou mnoha imitací, mimo jiné i pro Petřínskou rozhlednu.



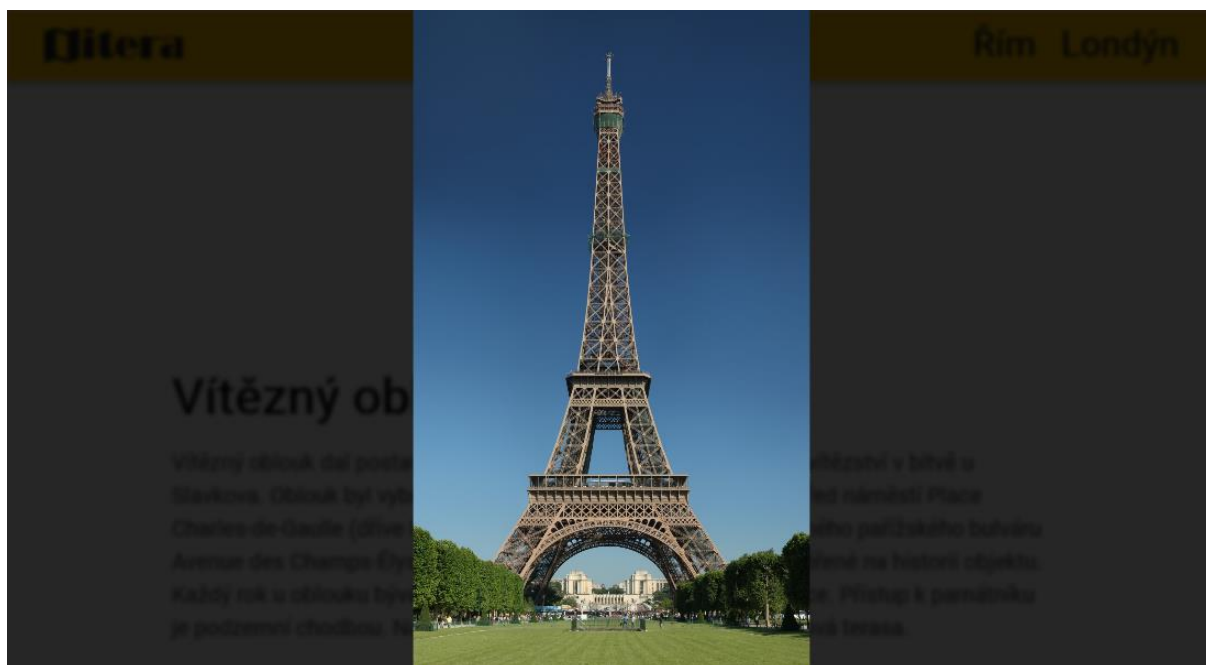
Obrázek 5: Stránka produktu

Na prvním obrázku druhé ukázky můžeme vidět stránku produktu (v tomto případě zájezdu do Paříže; textový obsah stránky byl pro účel obrázku zkrácen).

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. Šířka hlavního obsahu i s obrázky je

však omezena tak, aby řádky s textem nebyli příliš dlouhé, což by způsobovalo větší únavu očí při čtení obsahu.

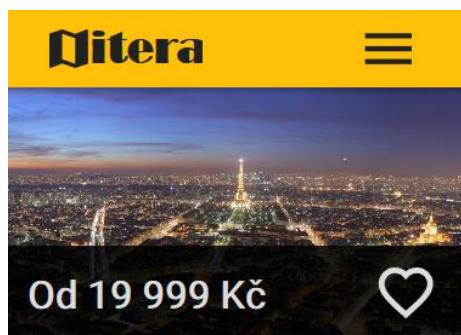
Fotografie je možno stisknutím levého tlačítka myši (popřípadě klepnutím prstu na dotykové obrazovce) přiblížit.



Obrázek 6: 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 na nejvyšší 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. Přiblížení je vyobrazeno na obrázku 6.





## Paříž

Věž byla předlohou mnoha imitací, mimo jiné i pro Petřínskou rozhlednu.



Obrázek 7: Stránka produktu zobrazená na telefonu – část 1



## Louvre

Louvre je muzeum umění v Paříži, které patří k největším na světě. Na ploše 72 tisíc m<sup>2</sup> vystavuje přes 35 tisíc exponátů (z celkové sbírky asi 554 tisíc) od pravěku po 19. století a ročně je navštíví přes 8 milionů diváků. Sídli v palácovém komplexu Palais du Louvre, bývalém sídle francouzských králů. Leží v 1. obvodu na pravém břehu řeky Seiny.



Obrázek 8: Stránka produktu zobrazená na telefonu – část 2

Na obrázcích 7 a 8 se nachází stránka produktu zobrazená na mobilním telefonu (text stránky v obrázku 7 částečně zkrácen).

V horní části můžeme opět vidět mobilní verzi horního menu, kdy seznam s odkazy je ve výchozím stavu skryt. Hlavní část stránky s obsahem využívá celé šířky obrazovky. Obrázky však plnou šíři vyžívají pouze tehdy, není-li daný obrázek příliš vysoký (aby nezabíral příliš místa na obrazovce). Text je lehce odsazen od kraje.

## Závěr

Ve své 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 laptotech, 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 mé maturitní práce 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](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](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](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](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: README [online]. c2017 [citováno 17. 03. 2018]. Dostupný z WWW: <https://github.com/google/guetzli/blob/master/README.md>

- [12] Wikipedia, The Free Encyclopedia: Name mangling [online]. c2018 [citováno 24. 03. 2018]. Dostupný z WWW: [https://en.wikipedia.org/w/index.php?title=Name\\_mangling&oldid=831981276](https://en.wikipedia.org/w/index.php?title=Name_mangling&oldid=831981276)
- [13] Wikipedie: Otevřená encyklopedie: Verzování [online]. c2018 [citováno 25. 03. 2018]. Dostupný z WWW: <https://cs.wikipedia.org/w/index.php?title=Verzov%C3%A1n%C3%AD&oldid=15957985>
- [14] Node.js [online]. c2018 [citováno 25. 03. 2018]. Dostupný z WWW: <https://nodejs.org/en/>