Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologii

Bakalářská práce

Vliv vybraných vlastností a nástrojů CSS na rychlost načítání webových stránek

Jiří Patejdl

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Jiří Patejdl

Informatika

Název práce

Vliv vybraných vlastností a nástrojů CSS na rychlost načítání webových stránek

Název anglicky

The effect of selected CSS properties and tools on the web page loading

---

Cíle práce

Tato bakalářská práce se bude zabývat problematikou vykreslování webových stránek na straně klienta. Jejím hlavním cílem je zhodnotit vliv a identifikovat limity vybraných vlastností, nebo nástrojů CSS na načítání webových stránek a aplikací.

Dílčími cíli této bakalářské práce jsou:

- Vybrat vlastnosti a nástroje CSS pro další analýzu.

- Vytvořit nebo zvolit vhodné testovací webové stránky.

- Navrhnout a provést experimentální měření výkonu načítání testovacích stránek.

Metodika

Teoretická část se bude zabývat pravidly a doporučeními pro psaní kaskádových stylů, které slouží ke správné optimalizaci webových stránek.

V praktické části bude využito již dostupných vývojářských nástrojů, například těch, které v sobě mají zabudované jednotlivé prohlížeče. Těmito nástroji bude zhodnoceno, jak konkrétní CSS prvky zpomalují a ovlivňují načítání webové stránky a vytíženost uživatelského zařízení. Měřen bude vždy jeden prvek kaskádových stylů. Měření bude pokaždé opakováno tak, aby došlo k eliminaci vedlejších vlivů, jako je nestabilní internetové připojení, nebo aktuální vytížení procesoru jinými procesy.

Na základě zjištěných poznatků z teoretické části a následnému vyhodnocení výsledků z části praktické budou formulovány závěry práce.

Doporučený rozsah práce

40 – 50 stran

Klíčová slova

CSS, HTML, Vykreslování webových stránek, Rychlost načítání webových stránek

---

Doporučené zdroje informací

ATTARDI, Joe. Modern CSS: Master the key concepts of CSS for modern web development. New York : Apress, 2020.

CASTRO, Elizabeth. HTML5 a CSS3: Názorný průvodce tvorbou WWW stránek. Brno: Computer Press, 2007. ISBN: 978-80-251-1531-2.

DUCKETT, Jon. HTML & CSS: design and build websites. Indianapolis, IN: Wiley, [2011]. ISBN 1118008189.

Firefox source tree documentation . Firefox Source Tree Documentation – Firefox Source Docs documentation [online]. [Accessed 20 May 2022]. Available from: https://firefox-source-docs.mozilla.org/index.html

GASSTON, Peter. CSS3. Přeložil Ondřej BAŠE. Brno: Computer Press, 2016. ISBN 978-80-251-4641-5.

Mdn.dev. mdn.dev [online]. [Accessed 20 May 2022]. Available from: https://mdn.dev/

MICHÁLEK, Martin. CSS: MODERNÍ LAYOUT. 1. 2022. ISBN 978-80-88253-07-5.

Předběžný termín obhajoby

2022/23 LS – PEF

Vedoucí práce

Ing. Jan Masner, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 14. 7. 2022

doc. Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 27. 10. 2022

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 14. 02. 2023

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Vliv vybraných vlastností a nástrojů CSS na rychlost načítání webových stránek" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne _____

Poděkování

Rád bych touto cestou poděkoval panu Ing. Janu Masnerovi Ph.D. za profesionální vedení mé bakalářské práce.

Vliv vybraných vlastností a nástrojů CSS na rychlost načítání webových stránek

Abstrakt

Cílem této bakalářské práce je zjistit, jaký mají vliv různé vlastnosti a nástroje CSS na načítání webových stránek.

Teoretická část se zabývá všeobecnými pravidly a vlastnostmi pro psaní CSS a na tyto zjištěné poznatky navazuje teoretická část.

Vybrané CSS vlastnosti byly přidány na experimentální webovou stránku, na které následně probíhalo měření rychlosti načítání a hardwarové náročnosti CSS animací s vybranými vlastnosti. Výsledky měření pak byly porovnány s původními výsledky.

Měření probíhalo také pro počet CSS soborů, kde bylo zjištěno, že se vyplatí na malé webové prezentaci používat jeden, oproti komponentovému CSS soborů.

Z výsledků měření tak bylo zjištěno, zdali mají vybrané vlastnosti a počet CSS soborů vliv na načítání webové stránky. A byla sepsána doporučení pro vývojáře.

Klíčová slova: HTML, CSS, Webová Stránka, Vykreslování, Načítání, Optimalizace, Vývojářské Nástroje, Analýza

The effect of selected CSS properties and tools on the web page loading

Abstract

The aim of this bachelor's thesis is to find out how different CSS properties and tools affect the loading of web pages.

The theoretical part deals with general rules and properties for writing CSS and the theoretical part builds on these findings.

The selected CSS properties were added to an experimental web page, which was then used to measure the loading speed and hardware performance of CSS Animations with the selected properties. The measurement results were then compared with the original results.

Measurements were also run for the number of CSS files, where it was found that it was more profitable to use one CSS file on a small website, compared to a component CSS files.

Thus, the measurement results were used to determine whether the selected features and the number of CSS files have an effect on the loading of a web page. And recommendations for developers were made.

Keywords: HTML, CSS, Web Page, Rendering, Loading, Optimization, Developer Tools, Analysis

Obsah

1 Úvod

Jedním z ukazatelů dobře optimalizované webové stránky může být čas, který na ní uživatel stráví a zdali se na stránku následně vrátí. V tomto případě hraje klíčovou roli správné a rychlé načítání, společně se správným zobrazením, čehož lze docílit správnou optimalizací, kdy jedním ze stěžejních kroků je například optimalizace kaskádových stylů. Pro spokojenost uživatele s webovou stránkou mohou být důležité mimo jiné dva faktory, a to, jak rychle dokáže uživatel na stránce vyhledat informace a také za jak dlouho se mu tyto informace na stránce zobrazily. Pokud se stránka načítá příliš dlouho, uživatel zpravidla odchází hledat informace jinam a s velkou pravděpodobností se na stránku již nevrátí.

Spokojeností návštěvníka se zabývají vývojáři se specializací na „user experience“, tedy o uživatelskou zkušenost, která se snaží maximalizovat spokojenost a množství interakcí uživatele s webovou stránkou. UX a UI designeři se společně s developery snaží o maximální efektivitu, díky které lze předejít takzvanému „bounce efektu“, to je situace, kdy uživatel stránku navštíví, avšak ve velice krátkém časovém rozpětí jí zase opouští. Uživatelsky zpříjemnit webovou stránku je také možné pomocí CSS animací, které pomáhají udržet pozornost uživatele.

Zrychlení načítání stránky je tedy esenciální a tento fakt si uvědomuje většina prodejců a majitelů e-shopů po celém světě, čím je stránka přehlednější a uživatelky přívětivá, tím je větší šance, že se uživatel na danou stránku vrátí, a tak zvyšuje „conversion rate“ čili poměr mezi uživateli, kteří stránku navštívili a kteří si něco koupili. Kvůli tomuto důvodu se vývojáři začali soustředit také na to, jak uživatele na stránce udržet a jak mu co nejvíce zpříjemnit čas strávený na stránce.

2 Cíl práce a Metodika

Tato bakalářská práce se zabývala problematikou vykreslování webových stránek na straně klienta. Jejím hlavním cílem bylo zhodnotit vliv a identifikovat limity vybraných vlastností, nebo nástrojů CSS na načítání webových stránek a aplikací.

Dílčími cíli této bakalářské práce byly:

- Vybrat vlastnosti a nástroje CSS pro další analýzu.
- Vytvořit nebo zvolit vhodné testovací webové stránky.
- Navrhnout a provést experimentální měření výkonu načítání testovacích stránek.

2.1 Metodika

Teoretická část se zabývala pravidly a doporučeními pro psaní kaskádových stylů, které slouží ke správné optimalizaci webových stránek.

V praktické části bylo využito již dostupných vývojářských nástrojů, například těch, které v sobě mají zabudované jednotlivé prohlížeče. Těmito nástroji bylo zhodnoceno, jak konkrétní CSS prvky zpomalují a ovlivňují načítání webové stránky a vytíženost uživatelského zařízení. Měřen byl vždy jeden prvek kaskádových stylů. Měření bylo pokaždé opakováno tak, aby došlo k eliminaci vedlejších vlivů, jako je nestabilní internetové připojení, nebo aktuální vytížení procesoru jinými procesy.

Na základě zjištěných poznatků z teoretické části a následnému vyhodnocení výsledků z části praktické byly formulovány závěry práce.

3 Teoretická východiska

V teoretických východiskách je zapotřebí specifikovat základní pojmy, které se týkají tohoto tématu. Tato sekce seznámí čtenáře se základními tématy, které je důležité znát pro správné pochopení této problematiky.

3.1 World Wide Web (WWW)

Jeho autorem je Tim Berners-Lee. Jinak je také nazýván jako W3 nebo The Web. World Wide Web a internet není zdaleka to samé. Zatímco internet je globální systém propojených počítačových sítí, WWW je propojený systém webových stánek a dokumentů, které jsou uloženy na různých webových serverech připojených k internetu, které lze zobrazit za pomocí webových prohlížečů, HTTP protokolu, odkazů (hyperlinků) a URL. (1)

3.2 Webový prohlížeč

Webový prohlížeč je nástroj, který umožnuje zobrazení webové stránky na uživatelově zařízení za využití HTTP protokolu, který definuje, jak jsou data přenášena na webu. Tato data musí být zobrazena v konzistentním formátu, aby uživatel viděl stejný obsah, z jakéhokoliv prohlížeče. Vzhledem k faktu, že jednotlivé prohlížeče jsou vyvíjeny nezávislými firmami, mohou určité informace interpretovat jinak, než bylo původně zamýšleno a lze tak dosáhnout špatné či zhoršené funkcionality webové stránky. Z tohoto důvodu se vývojáři musí držet takzvanými webovými standarty pro vývoj webových stránek. (2)

3.2.1 Vykreslovací jádro

Každý prohlížeč v sobě má zabudovaný kus softwaru, který se nazývá vykreslovací jádro (Rendering Engine), které slouží pro transformaci HTML a XML dokumentů, CSS kódu a dalších částí webové stránky do interaktivní reprezentace na uživatelově zařízení. (3)

3.3 HyperText Markup Language (HTML)

Jedná se o celosvětově nejpoužívanější značkovací jazyk. Tim Berners-Lee, autor WWW vytvořil tento jazyk za účelem šíření textových dokumentů po síti. Tyto dokumenty se dají zobrazit ve webových prohlížečích. (4)

HTML udává dokumentu takzvanou „kostru“ webové stránky, která nám určí rozpoložení jednotlivých elementů na stránce. Pomocí čistého HTML však nelze udělat stylizaci stránky, k tomu je zapotřebí využití kaskádových stylů. Lze manipulovat s HTML dokumenty a DOM pomocí skriptovacího jazyku jménem JavaScript. (5) (6)

3.4 Kaskádové styly (CSS)

V angličtině jsou známy pod pojmem Cascading Style Sheets neboli CSS. Jejich funkcí je stylizace dokumentů psaných ve značkovacích jazycích jako je například HTML a XML. Kaskádové styly tedy určují, jak budou jednotlivé elementy a části dokumentu vypadat na vykreslené webové stránce. (7)

3.4.1 Limitace a nevýhody CSS

Díky CSS lze stylizovat webovou stránku, skoro vše tak jde nastylizovat dle vývojářových představ, avšak CSS je limitováno schopnostmi jednotlivých webových prohlížečů, kdy některá stylizace je podporována jen v určitých verzích jednotlivých prohlížečů. Může se tak stát, že stylizace bude fungovat například v prohlížeči Google Chrome a v prohlížeči Safari nikoli, nebo nebude vypadat tak, jak bylo původně zamýšleno. Pro podmíněné stylizování je nutno využití například JavaScriptu, který je schopný sledovat události na stránce. CSS může být ze začátku komplikované pro začínající vývojáře. (8)

3.4.2 CSS Layout

Neboli také rozložení elementů na webové stránce. Rozložení je v režii samotného vývojáře, avšak existují základní layouty, na které jsou uživatelé zvyklí a tím pádem jim lze zpříjemnit uživatelskou zkušenost. Obvykle se na každé stránce nachází navigace, která je buď nahoře na stránce nebo na levé straně, obsah stránky dělený do sekcí a patička. (9)

3.4.3 Responsivní CSS

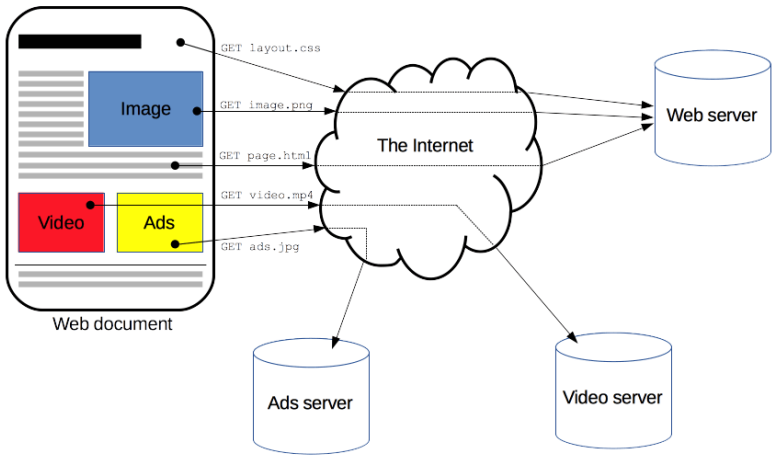
Aby byla stránka responzivní čili správně zobrazována na jakémkoli zařízení, je třeba využití základních CSS vlastností. Takovými vlastnostmi jsou například media queries, flexbox a grid. Kdy pomocí media queries lze specifikovat, jednotlivé stylizace na různých velikostech zařízení. Lze tak například určit, že v momentě, kdy jsme na zařízení, které má šířku obrazovky například 500pixelů a více, změní se šířka elementu na například 200px.

Dalšími vlastnostmi jsou „grid“, ten je využíván v případě, že je třeba dostáhnout specifického mřížkového rozložení na stránce a „flexbox“ pro dynamický počet řádků či sloupců. (6)

3.5 Hyper-Text Transfer Protocol (HTTP)

HTTP je client-server protokol, sloužící pro získávání dat a souborů od webových serverů. Veškeré požadavky jsou tedy vykonávány uživatelem, nebo spíše prohlížečem. Celkový dokument je tak seskládán z několika pod-dokumentů, jako je například text, rozložení stránky, obrázky a videa. Client (uživatel) a Server mezi sebou komunikují, Client odesílá požadavky (requests) na server a server mu nazpět odesílá odpovědi (responses), jak lze vidět na obrázku 1. (10)

Postupem času však HTTP protokol přechází na novou zabezpečenou verzi HTTPS, díky kombinaci HTTP a společně s TLS a SSL. Ze začátku byl však využíván pouze stránkami, které pracovaly s citlivými daty, jako například banky. Dnes je již tento protokol standardem. (11)



Obrázek 1 – Funkce HTTP protokolu (5)

3.6 Webová Stránka

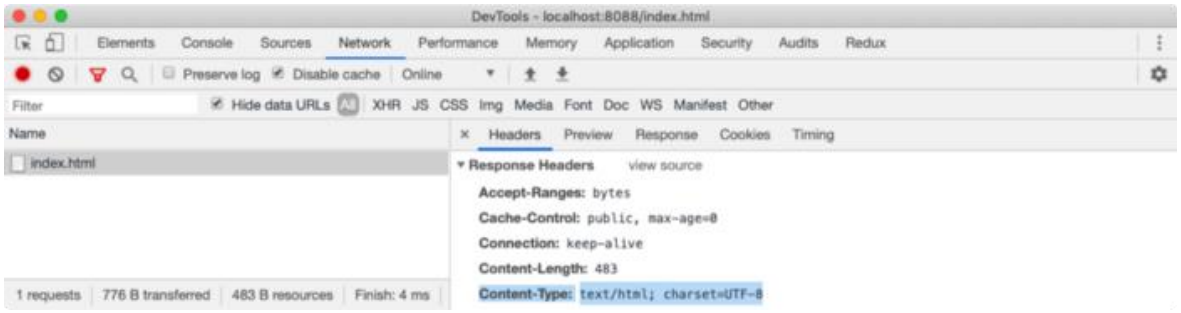
Webová stránka je dokument, který je napsaný ve značkovacím jazyce HTML, a následně stylizován pomocí kaskádových stylů. Na webovou stránku se lze dostat pomocí webového prohlížeče po zadání takzvaného URL čili jedinečné adresy dané stránky, to znamená, že dvě stránky nikdy nemohou mít stejnou URL adresu. Každá tato stránka může mít více podstránek, které jsou součástí této hlavní stránky. (12)

Webové stránky dělíme podle způsobu chování na dynamickou a statickou. Kdy na statické stránce uživatel nemůže vykonat žádnou akci, která by jakýmkoliv způsobem ovlivnila chování či vzhled stránky. Běžně jsou tyto stránky psány pouze za použití HTML, CSS a jednoduchého JS. Oproti tomu na dynamických stránkách může uživatel interagovat s určitými prvky, které se na stránce nacházejí, jako je například kontaktní formulář, registrace, přihlášení, či nákupní košík. Pro dosáhnutí těchto funkcionalit je tak potřeba využití například PHP a JS. (12)

3.6.1 Načítání Webové Stránky

Kdykoliv developer vytváří webovou stránku je potřeba se zamyslet nad zkušeností, kterou uživatel zažije při interakci se stránkou neboli také User Experience (UX). Stránka se může načítat pomalu, pokud zprvopočátku načítá zbytečné soubory, které nejsou potřeba pro správný chod stránky. Příkladem jsou obrázky, velké soubory a CSS kód, který není zprvopočátku nutný. Pro optimalizaci je třeba pochopit, jak prohlížeč zpracovává data, a jak tyto data následně zobrazuje.

V momentě, kdy prohlížeč odešle požadavek pro zobrazení HTML dokumentu, server odpoví v binární soustavě. V případě, že byl dokument převeden na kódu pomocí UTF-8 viz obrázek 2, využije prohlížeč tuto informaci a následně převede binární formát do textové podoby. Pokud by chyběla informace, zda byl použit formát UTF-8, prohlížeč by nebyl schopen zobrazit data. (13)

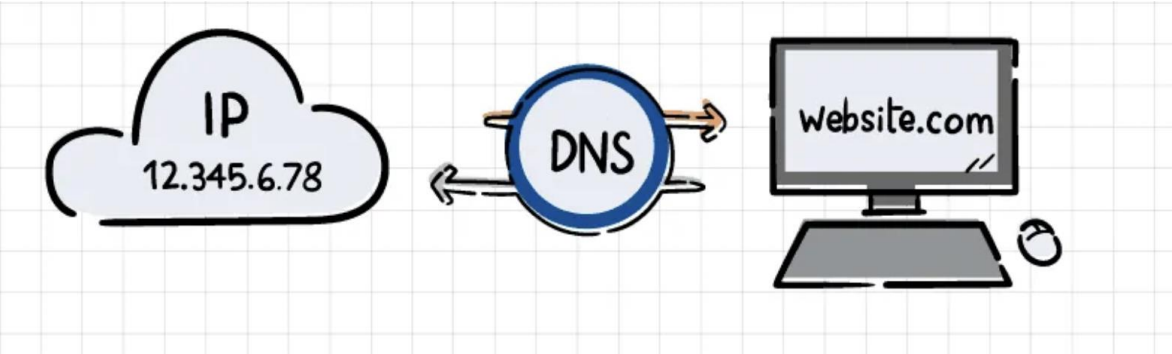


Obrázek 2 – Jak prohlížeč dostává informaci o UTF-8 (13)

3.6.2 Domain Name System (DNS)

Každá stránka má svou jedinečnou IP adresu, aby byla lépe rozpoznatelná webovým serverem. Uživatel běžně zadává do vyhledávače takzvanou doménu, jako je například „www.seznam.cz“ avšak webový prohlížeč tuto adresu vyhledává pomocí Internetového Protokolu (IP), daná adresa by tedy mohla vypadat například takto „77.75.76.3.“ (14)

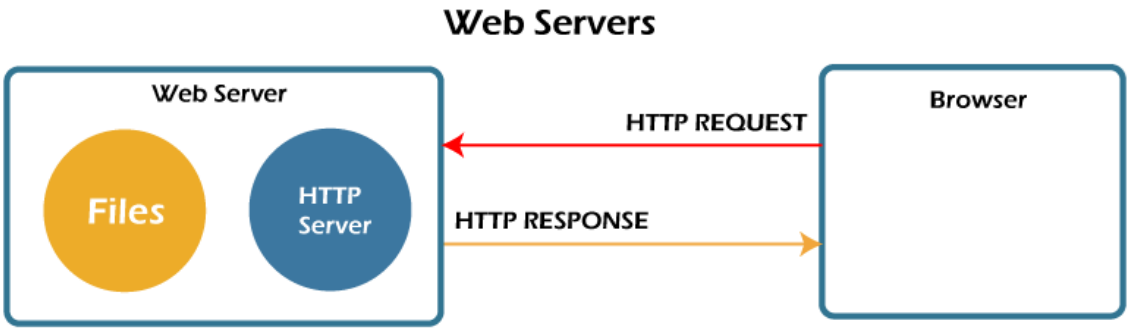
Aby si uživatel nemusel tuto složitou číselnou adresu pamatovat, vznikl DNS, který tyto číselné adresy přetváří na domény. Lze si ho tedy představit jako telefonní seznam pro internet, který spojuje IP adresy (telefonní čísla) s názvy domény (jmény), jak lze vidět na obrázku 3. (14)



Obrázek 3 – Domain Name System (15)

3.6.3 Webový Server

Je kombinací softwaru a hardwaru. Využívá HTTP protokolu pro komunikaci s uživatelem na WWW. Uživatelé vysílají požadavky na server o zobrazení webové stránky, server jim odpovídá a posílá data to jejich zařízení, která jsou následně přečtena a zobrazena webovým prohlížečem na uživatelské straně, viz. obrázek 4. Mimo protokolu HTTP, je webový server schopný využívat také protokolů SMTP (Simple Mail Transfer Protokol) a také FTP (File Transfer Trotokol), které jsou využívány pro elektronickou poštu a pro přesun souborů. (16)

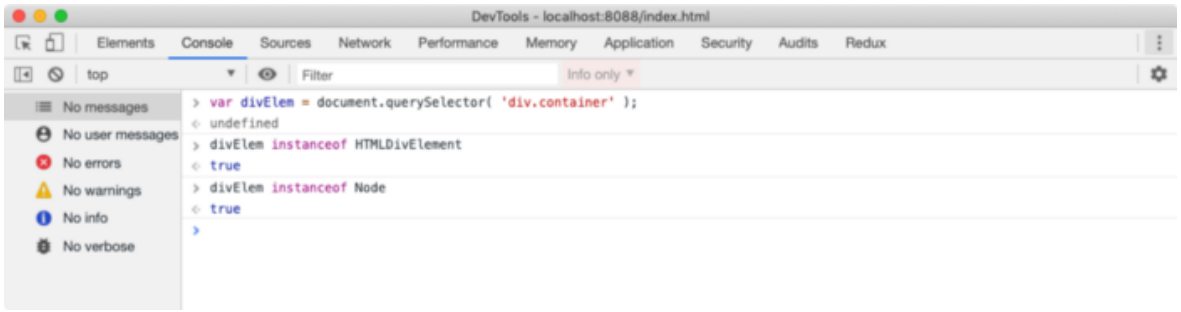


Obrázek 4 - Funkčnost Webového Serveru (17)

3.6.4 Document Object Model (DOM)

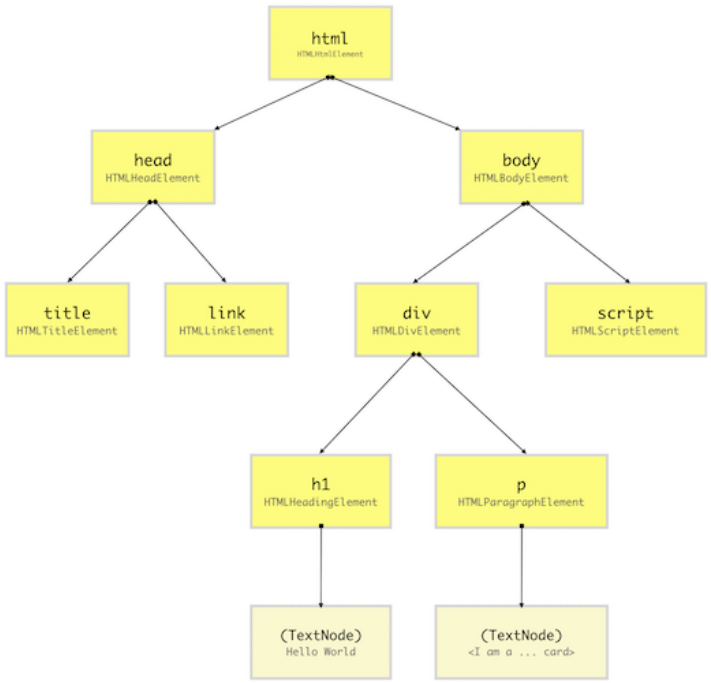
Nejprve je potřeba pochopit, jak funguje Document Object Model, zkráceně DOM. DOM definuje logickou strukturu dokumentu a určuje, jakým způsobem k němu bude přistupováno a zacházeno. Je to API pro HTML a XML dokumenty.

Kdykoliv prohlížeč rozpozná HTML kód převede ho na JS objekt, který nazýváme „node“ (tj. uzel). Postupně takhle prohlížeč převede všechny části HTML stránky na „node.“ Například „div“ je vytvořen z „HTMLDivElement“ a dědí třídu „node,“ jak je uvedeno na obrázku č. 5. (13) (18)



Obrázek 5 – Ukázka, jak dom přeměňuje HTML prvky na nodes (13)

V momentě, kdy prohlížeč vytvoří uzly z HTML dokumentu, vytváří následně stromovou strukturu z těchto objektů. DOM tento strom se tvoří z nejvrchnějšího elementu (viz. obrázek 6), tím bývá HTML a dále se rozrůstá podle toho, jak je dokument strukturovaný. (13)

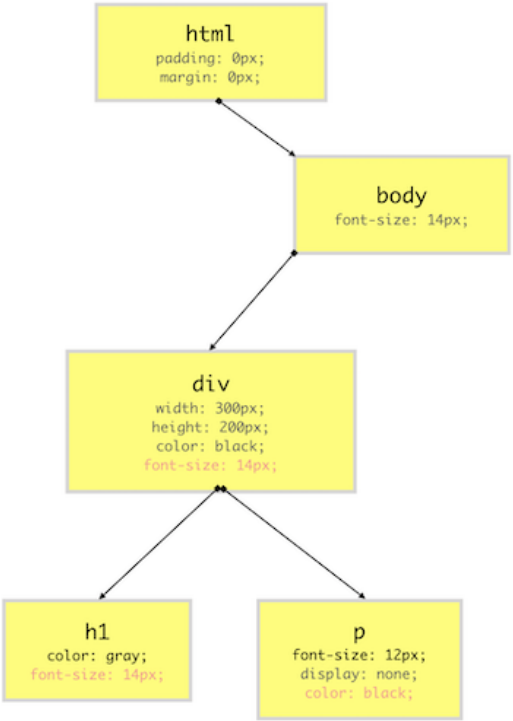


Obrázek 6 – Stromová struktura DOM (13)

3.6.5 CSS Object Model (CSSOM)

Na HTML stránce můžeme jednotlivé HTML elementy stylizovat pomocí Cascading Style Sheets (CSS), v překladu kaskádových stylů. Pomocí selektorů lze označit jednotlivé DOM elementy a změnit tak jejich výchozí hodnoty, jako je například barva, font a velikost. Takto lze měnit styly hned několika způsoby. Připojením CSS dokumentu, stylizování mezi <style> tagy v html dokumentu, inline či pomocí JS. (13) (18)

Poté co prohlížeč zkonstruuje DOM, začne číst CSS ze všech možných zdrojů a vytvoří tak CSSOM, což je stromová sktruktura, připomínající DOM, jak lze vidět na obrázku 7. Každý uzel v této struktuře nese informaci o tom, jak bude je DOM element stylizován, CSSOM strom však neobsahuje elementy, které nemohou být nijak vidět na stránce, jako jsou <script>, <title>, <meta>. (13) (18)

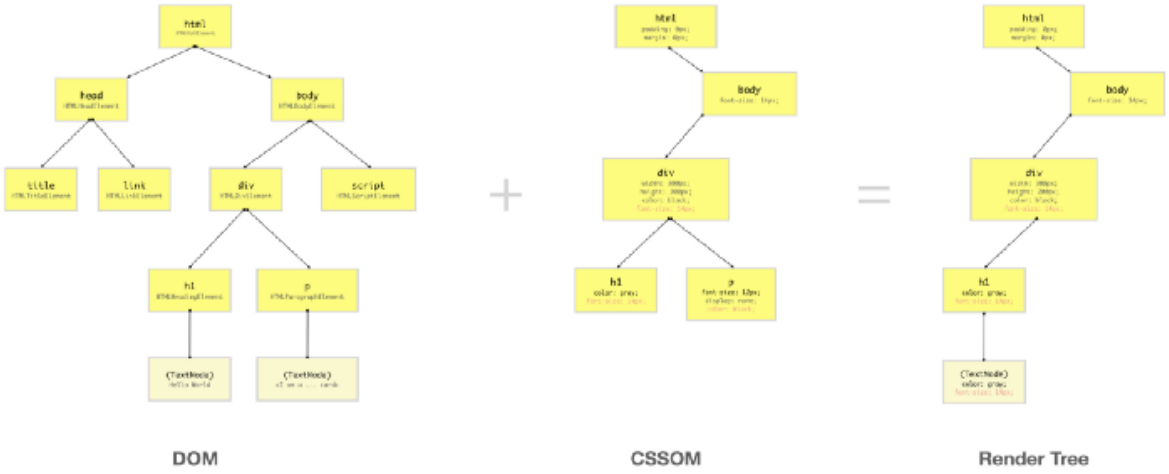


Obrázek 7 – Stromová struktura CSSOM (13)

3.6.6 Render Tree

V překladu vykreslovací strom je struktura, která kombinuje stromové struktury DOM a CSSOM. Prohlížeč následně vypočítá rozložení každého viditelného elementu a vykreslí je na obrazovce. Webová stránka se vykreslí pouze v případě, že je vytvořena stromová struktura CSSOM i DOM. Render Tree zobrazí pouze elementy, které mají rozměr, pokud bude mít element například „display: none“ má tak rozměry 0px a 0px, nebude tak

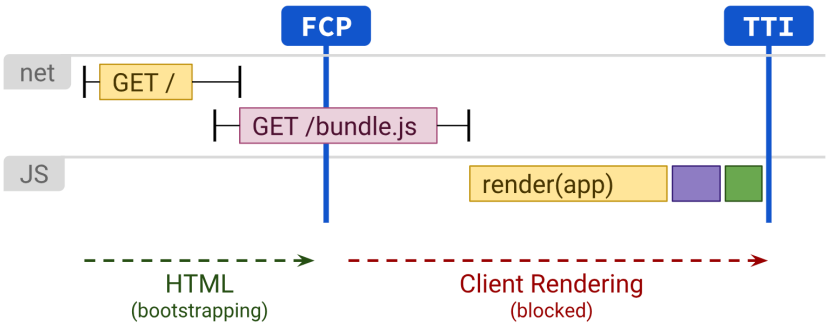
zobrazen v renderovacím stromu. Oproti tomu při element s „visibility: hidden“, nebo „opacity: 0“ budou na stránce vykresleny, vzhledem k tomu, že jsou zobrazeny v Render Tree. Příklad Render Tree lze vidět níže, na obrázku 8. (13) (18)



Obrázek 8 – Render Tree (13)

3.6.7 Client-Side Rendering (CSR)

Při vykreslování na straně uživatele většina akcí, děje na uživatelské straně v prohlížeči pomocí JavaScriptu. Veškerá logika je počítána prohlížečem, nikoli serverem. CSR může být komplikací například pro mobilní zařízení. Příklad průběhu SSR lze vidět na obrázku 9. (19)



Obrázek 9 – Client-Side Rendering (19)

3.7 Metodiky psaní CSS

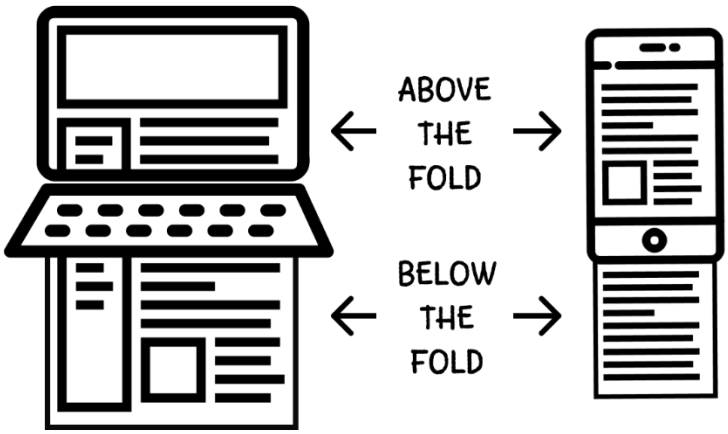
S každým velkým projektem je spojeno také jeden či více obsáhlých CSS souborů. V tomto případě je přehlednost daných souborů klíčovou částí pro úspěšný projekt. Ať už je projekt vytvářen v týmu, či jednotlivcem, je důležité držet se psaných pravidel pro psaní CSS, pokud má projekt dokumentaci pro psaní CSS musíme se jí vždy držet, jde nad naše

osobní preference. U kaskádových stylů často platí, že není jeden daný postup, jak se s danou problematikou vypořádat, je proto důležité, aby styl psaní byl konzistentní. (20)

V dnešní době již nemusí mít každý developer svá vlastní pravidla pro psaní kaskádových stylů. Pro správnou optimalizaci a budoucí práci na projektu je tedy doporučováno vybrat si jeden z již zaběhlých a ověřených způsobů, jak strukturovaně psát CSS. (20)

3.7.1 Kritické CSS

Prohlížeč musí stáhnout a zpracovat CSS dokumenty před tím, než uživateli zobrazí stránku, pokud je dokument moc veliký v kombinaci se špatnou kvalitou internetu zásadně zpomalí proces načítání webové stránky. Kritické CSS je technika, která využívá takzvané „above-the-fold“ a „below-the-fold“ z důvodu, aby se nám stránka zobrazila v co nejrychlejším čase. Above-the-fold je obsah, který uživatel vidí při načtení stránky, před tím, než začne scrollovat dolů. To, co uživatel nevidí při načtení se nazývá „below-the-fold“ viz. obrázek 10. (21)



Obrázek 10 – Příklad zobrazení Above / Below the fold (2)

Kritické CSS se používá z důvodu, kdy při velkých CSS souborech se CSS stává blokátorem pro správné načtení stránky a nastává tak bílá obrazovka. Dá se to obejít tím, že CSS načteme asynchronně pomocí JS a také tím, že umístíme nezbytné CSS do <head> tagu, jako je tomu na obrázku 11. (22)

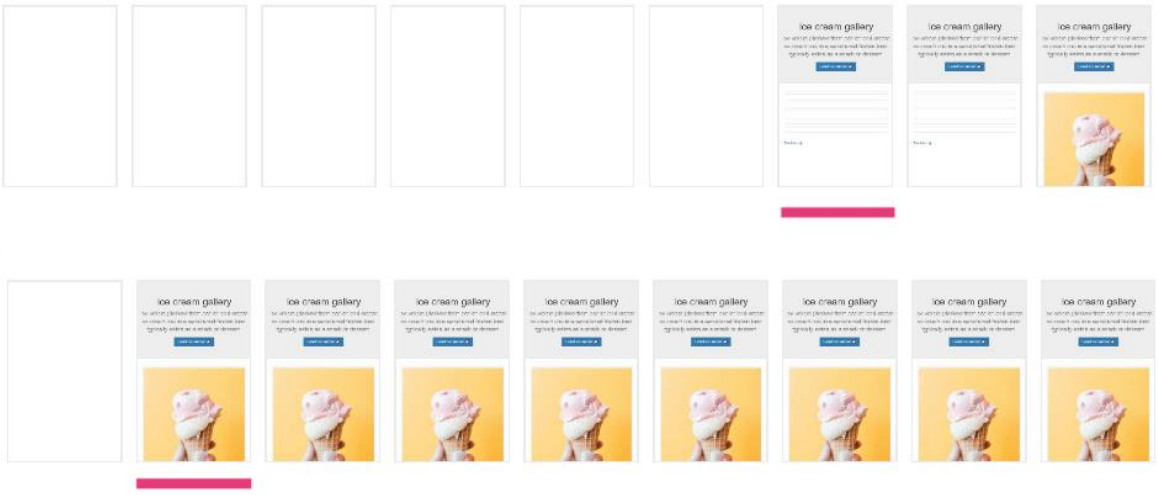
```
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shr
  <meta name="description" content>
  <title>Ice Cream gallery ·</title>

  <style>
    @-ms-viewport{width:device-width}html{font-family:sans-serif;-webkit-
  </style>

  <link rel="preload" type="text/css" href="https://maxcdn.bootstrapcdn.c
  <noscript><link rel="stylesheet" type="text/css" href="https://maxcdn.b
  <script>!function(n){"use strict";n.loadCSS||(n.loadCSS=function(){});v
</head>
```

Obrázek 11 – Načítání nezbytného CSS v inline podobě (21)

Tímto způsobem je ošetřeno, že uživatel nečeká, na stažení datově objemného CSS souboru a namísto toho stáhne styl, ve kterém se nachází základní kostra stránky, či části webu, kterou uživatel uvidí jako první čili část „above-the-fold“, přičemž se následující obsah stáhne až později. Na obrázku 12 tak lze vidět rozdíly v načítání (21)

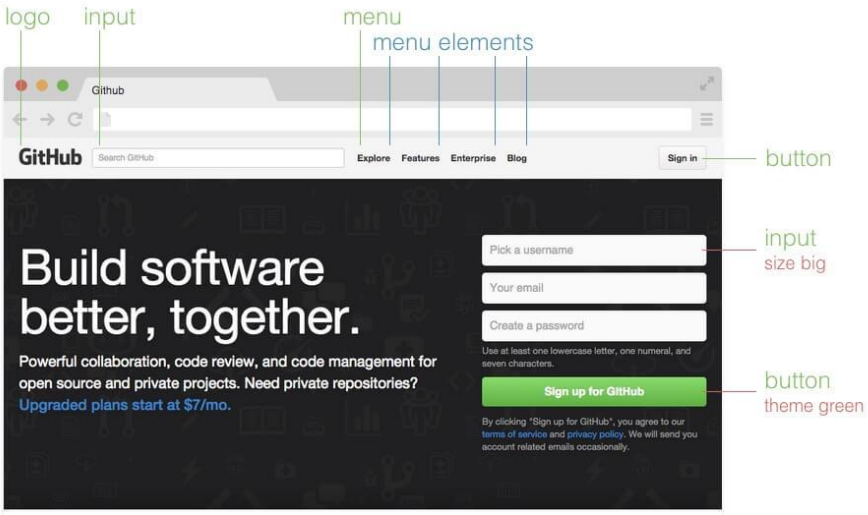


Obrázek 12 – Porovnání načítání webové stránky při klasickém zápisu CSS (nahoře) a té samé stránky při využití inline kritického CSS (dole) (21)

3.7.2 BEM

Na BEM můžeme pohlížet jako na plnohodnotnou metodiku pro organizaci CSS. Zkratka BEM znamená Block, Element, Modifier. Je vhodný pro psaní středních a velkých projektů. Pokud projekt používá BEM pojmenovávací konvenci tříd je to poznat již pouhým okem, vzhledem k nadměrnému používání pomlček a podtržítek. Někteří vývojáři považují

BEM spíše jako nadstavbu pro OOCSS. Na obrázku číslo 13 lze vidět, jak vypadá rozložení stránky v metodice BEM, kdy zelená barva znázorňuje Blok, modrá Element a červená Modifier. (20) (23)



Obrázek 13 – Ukázka rozložení stránky podle BEM (24)

Typ Třídy	**Způsob pojmenovávání**
Blok	.block
Element	.block__element
Modifikátor	.block--modifier
Modifikátor Specifického Elementu	.block__element--modifier

Tabulka 1 – Konvence pojmenování tříd v metodice BEM (23)

V tabulce číslo 1 lze vidět konvenční pojmenování tříd v této metodice. Příkladem využití BEM v navigaci na stránce můžete vidět na obrázku 14 a 15. Při tomto stylu zápisu je jasné, co jaká třída označuje či modifikuje. Nav označuje blok čili samostatný komponent a „nav—secondary“ s „nav—opacity“ jsou jeho modifikátory. (23)

```
<nav class="nav nav--secondary nav--opacity"></nav>
```

Obrázek 14 – Příklad zápisu tříd v metodice BEM (23)

Na obrázku níže lze vidět příklad navigačního baru za využití metodiky BEM.

```
<nav class="nav nav--secondary nav--opacity">
    <ul class="nav nav--secondary">
        <li class="nav__item"></li>
            <a href="/">Item Jedna</a>
        </li>
        <li class="nav__item nav__item--active"></li>
            <a href="/">Item Dva</a>
        </li>
    </ul>
</nav>
```

Obrázek 15 – Příklad zápisu navigačního na stránce pomoci metodice BEM (23)

Kdy je:

- .nav – blok (komponent)
- .nav-secondary – modifikátor (varianta komponentu)
- .nav__item – element (je uvnitř komponentu)
- .nav__item--active – modifikátor specifického elementu

3.7.3 OOCSS

OOCSS je zkratkou pro Object Oriented CSS (tj. objektově orientované CSS) a vznikl jako koncept pro psaní kódu, který vymyslela Niloce Sullivan. Cílem OOCSS je zajištění znovupoužitelnosti kódu a zároveň také zlepšení jeho údržby a redukce objemu CSS souborů. (25)

Hlavní myšlenkou OOCSS bylo rozdělení CSS na znovupoužitelné objekty, které mohou být využity kdekoliv na stránce. Kód je organizován do jednotlivých komponent, nemělo by se tedy stát, že by vývojář otevřel CSS soubor, který by obsahoval kód pro celou stránku či projekt. OOCSS je také možné využívat při práci s preprocesory jako je například SASS či LESS. (20)

OOCSS se drží pravidla rozdělení struktury a vzhledu. Strukturou je v tomto případě myšleno převážně to, co je pro uživatele „neviditelné“ například margin, padding, šířka a výška objektu a také udává, jak budou komponenty na stránce rozloženy. Vzhledem je myšlen například font, stín a barva. (26)

V případě, že je potřeba dvou stejně velkých tlačítek, kdy má každé jinou barvu, CSS kód by vypadal jako na obrázku 16 a 17, kdy je „. rozmer“ znovupoužitelnou třídou a lze ji tak

využít v projektu pokaždé kdy bude potřeba dosáhnout tohoto rozměru. Třída „. button“ modifikuje barvu komponentu.

```css
.button {
    background: white;

}

.button-2 {
    background: black;
}

.rozmer {
    width: 100px;
    height: 50px;
}
```

Obrázek 16 – Příklad správného použití OOCSS v CSS soboru (26)

```html
<button class="button rozmer">Tlačítko 1</button>
<button class="button-2 rozmer">Tlačítko 2</button>
```

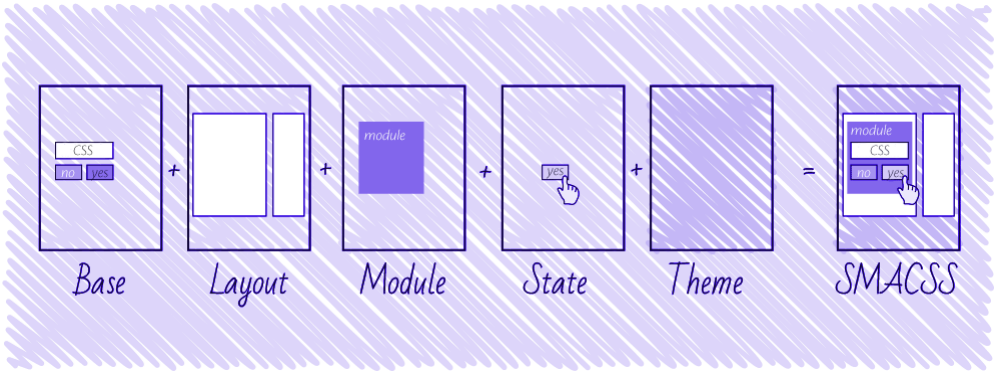
Obrázek 17 – Příklad správného použití OOCSS v HTML souboru (26)

Mezi hlavní výhody Objektově orientovaného CSS patří:

- Rychlost – vzhledem k menší opakovatelnosti v kódu dosáhneme efektivnějšího načítání, protože se tak lze vyvarovat datově objemným CSS souborům. (26)
- Efektivita – čím méně kódu se v souboru nachází, tím bývá přehlednějším. I v případě budoucích úprav a aktualizací je třeba upravit méně řádků. (26)

3.7.4 SMACSS

SMACSS je zkratka pro Scalabe and Modular Architecture for CSS. V překladu škálovatelná a modulární architektura pro CSS. Myšlenkou autora Jonathana Snooka bylo rozdělení CSS do pěti hlavních kategorii. Těmi jsou – Basic, Layout, Module, State a Theme. Lze vidět na obrázku 18. (27) (28)



Obrázek 18 – Rozdělení SMACSS do jednotlivých kategorii (29)

3.7.5 ACSS

Zkratka ACSS značí Atomic Cascading Style Sheets, jinak je také nazýváno Utility CSS. Autoři již v roce 2016 poukazovali na problémy spojené s velikostí jednotlivých CSS souborů, které obsahují spousty „mrtvého“ či duplicitního kódu, a těmto problémům se právě ACSS snaží předejít. (30)

Definice této metody zápisu CSS z článku napsaného Johnem Polacekem zní následovně – *„Atomic CSS is the approach to CSS architecture that favors small, single-purpose classes with names based on visual function.“* Funguje tedy na principu jednoúčelových tříd, které mají co nejkratší název, který by se měl podobat vizuální funkčnosti dané třídy. Při použití Atomic CSS by tedy soubor s kaskádovými styly mohl vypadat jako na obrázku 19. Lze zde vidět, že každá třída má pouze jednu funkcionalitu/stylizaci, od které se také odvíjí její název. (31)

```css
/* naming utility classes */
.relative {
  position: relative;
}
.mt10 {
  margin-top: 10px;
}
.pb10 {
  padding-bottom: 10px;
}
```

Obrázek 19 – Příklad zápisu Atomic CSS (31)

Existují dva hlavní způsoby, jak je možné psát ACSS, jedním z nich je staticky, kdy si developer vytváří třídy sám, popřípadě využije již předdefinovanou knihovnu tříd. Druhým způsob se nazývá „programmatic,“ kdy je třeba využít nástrojů pro automatické generování CSS kódu v návaznosti na jménu třídy, která je zapsaná v HTML dokumentu. Kdy na obrázku 20 lze vidět, jak by vypadal zápis v HTML a na obrázku 21 následně jak by byly automaticky vygenerované třídy v závislosti na HTML souboru. Příkladem tohoto nástroje je například Atomizer. (31) (32)

```html
<!-- Programmatic Atomic CSS Example -->
<div class="Bgc(#0280ae) C(#fff) P(20px)">Lorem ipsum</div>
```

Obrázek 20 – Zápis ACSS v HTML souboru s využitím nástroje Atomizer (31)

Obrázek 21 – Vygenerovaný CSS kód podle HTML souboru. (31)

Atomic CSS se ze začátku může zdát, jako by bylo třeba psát více kódu a tříd než při běžném psaní CSS. Je to pravda, vzhledem k tomu, že je třeba vytvořit třídu pro všechny potřebné barvy, margin, padding aj. Avšak pouze ze začátku, následně se pak rozsah potřebného CSS kódu pro vytvoření nového komponentu v HTML značně redukuje, vzhledem k tomu, že jakmile je jednoúčelová třída vytvořena, lze jí používat kdekoli v projektu. (31) Facebook v roce 2020 vydal novou verzi jeho Homepage, kdy tvrdí, že díky Atomic CSS byl zredukován potřebný rozsah CSS kódu o 80 %. (33)

3.8 Preprocesory CSS

Preprocesor je počítačový program, který transformuje CSS kód zapsaný jeho vlastní syntaxí do standardizovaného CSS zápisu, který je schopný přečíst prohlížeč. (34)
Mezi jejich hlavní výhody se počítají převážně zefektivnění a zjednodušení práce pro developery, zároveň tím lze dosáhnout méně obsáhlého CSS souboru, což značně urychlí načítání webové stránky. (35)

Pro využití preprocesoru ho je třeba nainstalovat přímo na webový server. Další možností je použít preprocesor, který lze nainstalovat přímo do developerského prostředí, ve kterém je stránka tvořena. Tím je myšleno například rozšíření pro SASS do vývojářské prostředí Visual Studio Code. Následně je pak nutno nahrát kompilovaný CSS soubor na webový server. (34)

3.8.1 SASS

Jeho zkratka, která znamená Syntactically Awesome Style Sheets. Tento preprocesor je nejpoužívanějším. (36) SASS je zároveň nejstarším preprocesorem, originálně vydán roku 2006 autory Natalie Weizenbaum a Hampton Catlin. (35)

Jeho hlavní výhodou je, že umožňuje vývojářům využívat programovací funkce jako jsou „if/else“, „for/while/loop“ a také nabízelo využití proměnných dříve, než je bylo možné využít v moderním CSS. Mimo jiné taky nabízí „nesting“ (tj. vnoření). (35)

SASS má hned dvě syntaxe. Dokumenty, využívající první, dnes již zastaralou syntaxi končí koncovkou „.sass“ s kompletní absencí složených závorek a středníků její kód vypadá následovně (viz. obr. 22.). (35)

```
/* Sass */

$primary-color: seashell
$primary-bg: darkslategrey

body
  color: $primary-color
  background: $primary-bg
```

Obrázek 22 – příklad zápisu zastaralé syntaxe preprocesoru SASS (35)

Nejnovější syntaxe však již využívá metody zápisu, s využitím středníků a složených závorek tak již připomíná syntaxi novodobého CSS, jak zle vidět na obrázku 23. Soubory psané v této metodě je nutné zakončit koncovkou „.scss“.

```
/* SCSS */

$primary-color: seashell;
$primary-bg: darkslategrey;

body {
  color: $primary-color;

  background: $primary-bg;

}
```

Obrázek 23 – příklad zápisu novodobé syntaxe preprocesoru SASS (35)

3.8.2 LESS

Leaner Style Sheets, byl vydán v roce 2009 čili tři roky po vydání SASS a také jím je z velké části tento preprocesor ovlivněn a nabízí stejné možnosti co SASS. LESS je oproti SASS JavaScriptová knihovna snažící se o rozšíření funkcionality klasického CSS. Ačkoliv LESS lze využít při renderování na uživatelské straně, samotní vývojáři doporučují využívat ho převážně k server-side renderingu. Příklad jeho zápisu lze vidět na obrázku 24. (35) (37)

```less
// LESS

#header {
  h1 {
    font-size: 26px;
    font-weight: bold;
  }
  p { font-size: 12px;
    a { text-decoration: none;
      &:hover { border-width: 1px }
    }
  }
}
```

Obrázek 24 – příklad zápisu kódu v preprocesoru LESS (38)

3.9 Počet CSS souborů

3.9.1 Jeden CSS soubor

V tomto případě načítá DOM z hlavičky souboru pouze jeden obsáhlý CSS soubor, který obsahuje veškerou stylizaci, nerozdělujeme zde do více souborů typografii, grid, flex a ani jednotlivé komponenty. Jeden CSS soubor je vhodným řešením pro menší jednostránkové, či vícestránkové webové stránky, kde se jednotlivé podstránky od sebe značně neliší, jeden CSS soubor je dobrým řešením. Příkladem je například blog, přestože je na jedné stránce několik stovek článků se stejným vzhledem, není nutné vytvářet další CSS soubory, které upravují vzhled každého tohoto článku. (39)

Značnou nevýhodou je však udržitelnost a zpětná editace tohoto souboru. Jeden soubor při špatné sémantice bývá po nějaké době značně nepřehledný i pro samotného autora a při nutném zásahu do CSS kódu se tak může developer potýkat s nepříjemnostmi, jako je špatná struktura a pojmenování tříd. (40)

3.9.2 Komponentové CSS

DOM načítá z hlavičky více CSS soborů (viz. obrázek 25), kdy každý obsahuje části stylů, podle volby developera. Developer má tím pádem možnost, rozdělit stylizaci na několik dílčích částí. Tato metoda je vhodná pro obsáhlejší webové stránky, kdy můžeme zvolit například využití komponentového CSS. Každý CSS soubor tak obsahuje specifickou stylizaci pouze pro daný komponent, značnou výhodou je tak udržitelnost a přehlednost kódu. Vzhledem k rozdělení daných komponentů se lehce pohybuje v jednotlivých souborech a jednotlivé požadované editace jednotlivých komponentů se tak značně urychlí.

Prohlížeč tedy posílá HTTP požadavek pouze v momentě, kdy se na stránce vyskytuje specifický komponent. (39) (40)

```
<link rel="stylesheet" href="./css/default-classes.css">
<link rel="stylesheet" href="./css/style.css">
<link rel="stylesheet" href="./css/navbar.css">
<link rel="stylesheet" href="./css/footer.css">
<link rel="stylesheet" href="./css/blog.css">
<link rel="stylesheet" href="./css/intro.css">
```

Obrázek 25 – Příklad implementace CSS stylů do hlavičky HTML (39)

Na obsáhlých webových stánkách je však ideálním řešením využití dvou, maximálně tří CSS souborů. Kdy v jednom jsou uchované výchozí styly pro často se opakující třídy a typografie a v ostatních se následně stylizují jednotlivé komponenty, které se mají lišit od výchozích stylů. (39)

3.10 CSS Animace

Díky CSS animacím lze změnit stylizaci elementu z jedné konfigurace na druhou. Animace se vytvářejí pomocí takzvaných „keyframes," které indikují, jak se má každá animace chovat v závislosti na čase. CSS Animace lze využívat i v kombinaci s JS, ve spoustě případů to však není potřeba. Animace můžou být jednorázové po načtení stránky, opakující se do zadaného počtu opakovaní, nebo také v nekonečné smyčce. (41)

Při tvorbě animací je tak třeba důkladně zvážit, jaké CSS vlastnosti budou pro danou animaci použity, vzhledem k faktu, že některé vlastnosti jsou náročnější oproti jiným. Pomoci prohlížeči lze také pomocí CSS vlastnosti „will-change," která prohlížeči sdělí, že se budou nějaké vlastnosti budou měnit, ještě před tím, se doopravdy změní, prohlížeč tak s touto informací počítá a animace tak můžou být plynulejší. (41)

Přestože animace mohou způsobovat problémy s interakcí na webové stránce, při správném provedení mohou výrazně zlepšit kvalitu stránky z UX hlediska, kdy je uživatel více pohlcen webovou stránkou, vzhledem k faktu, že díky animacím lze lépe upoutat uživatelovu pozornost, a tím pádem na stránce stráví větší množství času. Animace tak mohou zlepšit bounce rate a conversion rate. (42)

3.10.1 Transitions

Jedná o jednoduchý a méně pracný způsob, jak změnit CSS stylizaci z A na B. V tomto případě lze animovat například třídu za pomoci její pseudotřídy viz obrázek 26. Zde lze vidět, že třída „box" má nastavené nějaké vlastnosti, avšak v momentě, kdy uživatel

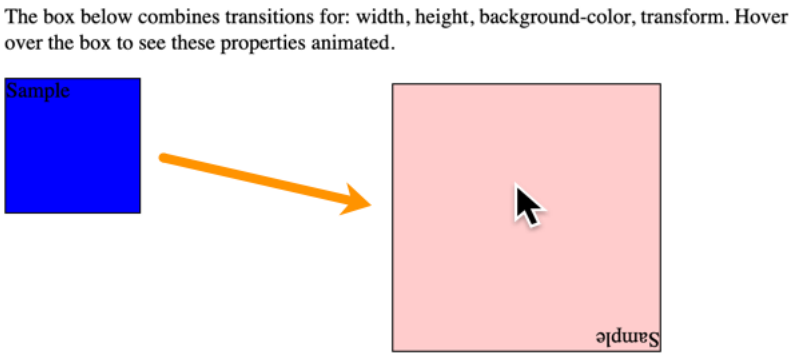
na element, který má třídu box umístí kurzor, stylizace se lineárně změní na vlastnosti nastavené níže. (43)

```css
.box {
  border-style: solid;
  border-width: 1px;
  display: block;
  width: 100px;
  height: 100px;
  background-color: #0000ff;
  transition: width 2s, height 2s, background-color 2s, transform 2s;
}

.box:hover {
  background-color: #ffcccc;
  width: 200px;
  height: 200px;
  rotate: 180deg;
}
```

Obrázek 26 – Ukázka CSS transition (43)

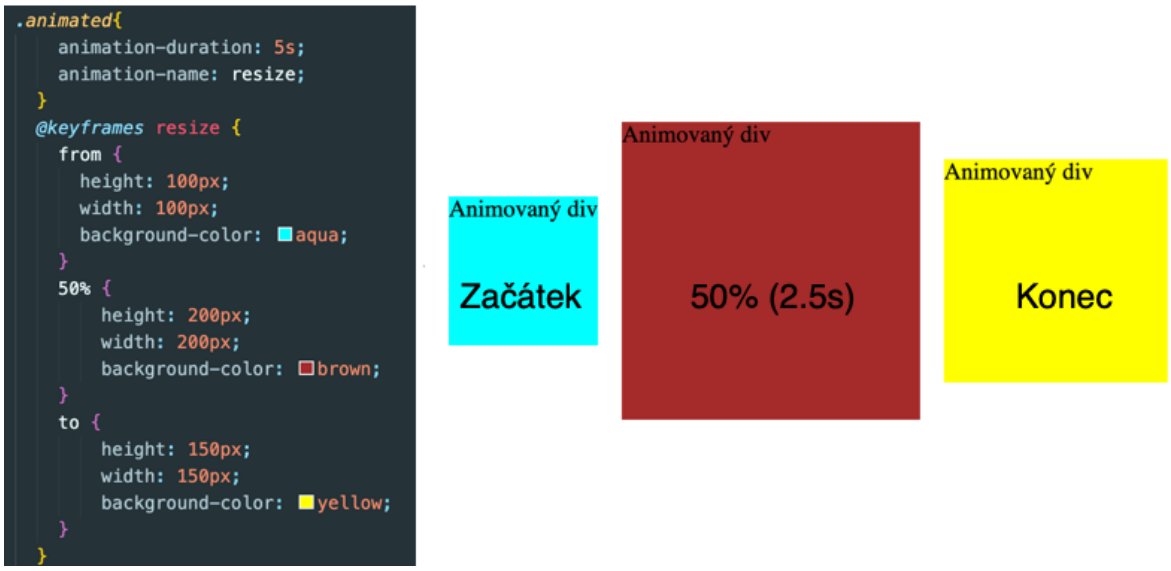
Jak lze vidět na obrázku 27, takto vypadá element s třídou „box“, před tím, než na něj umístíme kurzor a následně poté. Stylizace boxu se změní, po najetí kurzorem, jednotlivé vlastnosti se změnili za určený čas, v tomto případě jsou to dvě vteřiny. Využití jednotlivých vlastnosti a jejich hodnoty jsou však čistě na vývojáři. (43)



Obrázek 27 – Ukázka funkcionality CSS transition (43)

3.10.2 Keyframes

Keyframes se používají výhradně pro animování elementů na stránce. Díky této CSS funkci lze vytvořit jednu animaci, kterou lze pak používat vícekrát, keyframes lze tak využívat podobně, jako je tomu v CSS metodice OOCSS, kdy není nutné pokaždé vytvářet novou animaci pro každý element, který chceme animovat a nedochází tak k duplikaci kódu. (44)



Obrázek 28 – Příklad zápisu a funkcionality keyframes. (43)

3.10.3 Vliv animací na PagePerformance

CSS animace mají vliv na hodnotu rychlostního indexu, v ideálním případě co nejmenší. Pro nejlepší optimalizaci stránky by měly být využívány bez použití JavaScriptu, lze se tak vyvarovat nepříjemným problémům. Při použití CSS animací, které nejsou nijak komplexní by načítání stránky nemělo být nijak markantně ovlivněno. Při použití většího množství komplexních animací s náročnými CSS vlastnostmi lze však pozorovat zhoršení celkového hodnocení stránky. (45)

3.11 Měření načítání webové stránky

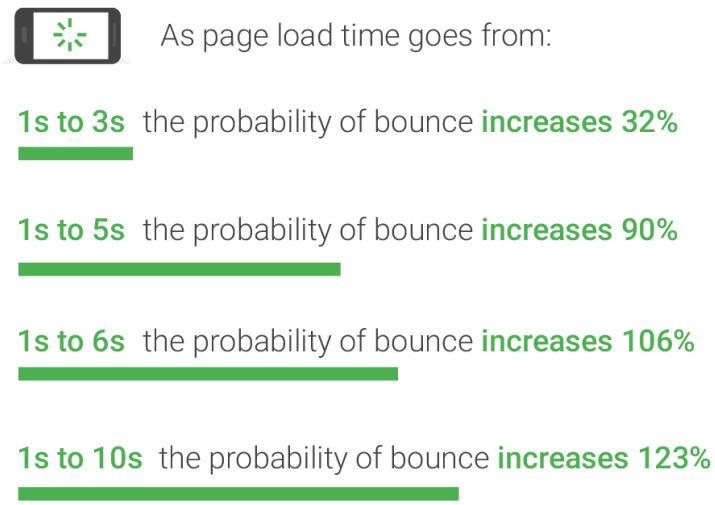
Načítání webové stránky lze měřit pomocí různých metrik a typů měření. Všeobecně platí, že pokud se stránka načítá dostatečně rychle, je pravděpodobnější, že uživatel dokončí konverzi na webové stránce čili v případě e-shopu zrealizují nákup. Čím rychleji se tedy stránka načte kompletně, tím lepšího dosáhne uživatelské zkušenosti. (46)

Existují taky studie, která dokazují, jak uživatel reaguje, když se stránka načítá určitou dobu. Mluvíme zde tak o hranicích:

- 0.1 sekundy – jedná se o ideální čas odezvy webové stánky, uživatel nepociťuje žádné zdržení, a ihned se stránkou interaguje. (46)
- 1 sekunda – Přesto že uživatel pociťuje zdržení, jeho uživatelská zkušenost se značně nezhorší. (46)

- 10 sekund – V tomto případě uživatel provádí během načítání i jiné úkony, ale s největší pravděpodobností stránku rovnou opouští a hledá jinou, s podobným obsahem. (46)

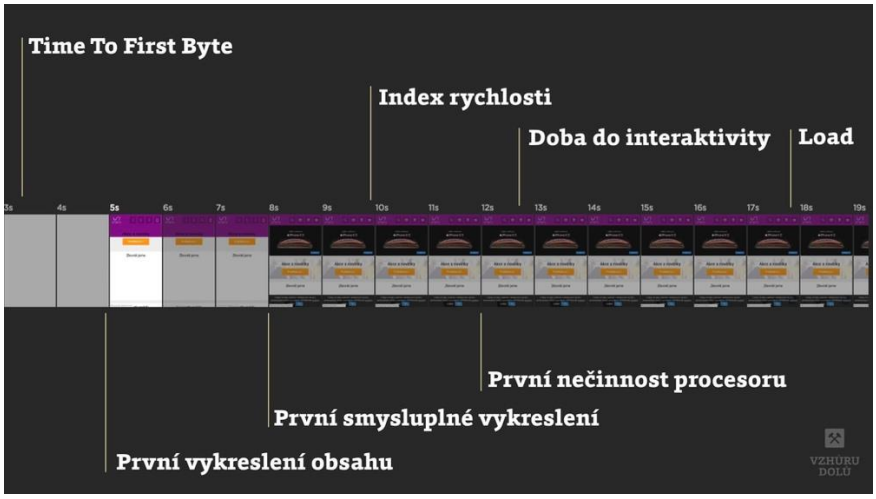
Můžeme zde hovořit o takzvaném „bounce.“ Dle Google se jedná o webovou analytiku, která značí, že uživatel navštívil pouze jednu stránku a následně ji opustil a dále s webovou stránkou neinteragoval, viz obrázek 29. (47)



Obrázek 29 – Pravděpodobnost "bounce" při časech načítání webové stránky. (46)

3.11.1 Metriky pro měření

Metriky nám uvádějí jednotlivé události, které nastávají při načítání každé webové stránky v čase. Na obrázku níže lze vidět tyto základní metriky pro měření a jejich hodnoty při načítání univerzitní webové stránky. Přestože těchto metrik existuje mnohem více, než je vidět zde na obrázku 30, pro účely této práce je dostačující zmínit šest nejdůležitějších. (48)



Obrázek 30 – Jednotlivé metriky pro načítání webové stránky zobrazené v čase. (48)

3.11.1.1 Time To First Byte (TTFB)

V překladu „doba do načtení prvního bytu“ udává, jak dlouho trvá prohlížeči stáhnutí prvního bytu webové stránky. Tato metrika mívá proměnlivou, kdy hraje roli rychlost odpovědi serveru, na kterém je webová stránka umístěna. Mimo jiné je ovlivněna rychlostí připojení samotného uživatele a také současnou vytížeností serveru. Přestože tato metrika nehraje zásadní roli v načítání webové stránky, její maximální hodnota by měla být 0,5 sekundy. (48)

3.11.1.2 DOM Content Loaded (DCL)

Metrika, která udává časovou hodnotu stažení a rozdělení celého HTML dokumentu, nečeká se na žádné další CSS, JS prvky a obrázky. Přestože není tato metrika na obrázku vyobrazena, objevuje se mezi prvním vykreslením obsahu (FCP) a prvním smysluplném vykreslením (FMP). (49)

3.11.1.3 Speed Index (SI)

Rychlostní index indikuje, jak dlouho trvá prohlížeči než, je obsah webové stránky zobrazen při načítání stránky. Je úměrný ostatním metrikám pro měření, kdy by se tak nemělo stát, že metrika FCP bude vysoká, zatímco SI bude mít poloviční hodnotu. (50)

3.11.1.4 First Contentful Paint (FCP)

V češtině „první vykreslení obsahu“ vzniká v momentě vykreslení prvního textu či obrázku, a to i těch, které jsou importovány za pomoci CSS s využitím vlastnosti „background.“ V případě že se načte FCP, uživatel tak získává pozornost nad událostmi, které se na stránce dějí, i přes to, že se stránka stále vykresluje a že v brzké době bude moci začít konzumovat obsah načítané webové stránky. (51)

Ideální hodnoty u FCP lze vidět v tabulce 2. V nástrojích PSI a Lighthouse se tato hodnota propisuje do celkového skóre s váhou 10 %. (51)

Hodnota LCP	Mobilní Zařízení	Desktop
Dobrá	≤ 1.8 sekundy	≤ 0.9 sekundy
Vyžaduje Zlepšení	≤ 3.0 sekundy	≤ 1.6 sekundy
Špatná	> 3.0 sekundy	> 1.6 sekundy

Tabulka 2 – Ideální hodnoty FCP (51)

3.11.1.5 Largest Contentful Paint (LCP)

Stala se náhradou pro nespolehlivou metriku First Meaninfull Pain, která značila čas do zobrazení prvního smysluplného obsahu neboli primárního obsahu. LCP tedy sleduje, kdy prohlížeč zobrazil největší prvek na stránce, který uživateli pomáhá vyhodnotit, zda je vyhledaná stránka užitečná. (52)

LCP tak vyhledává blokové prvky s CSS hodnotou „display: block“ které obsahují text, obrázky, elementy které mají v CSS „url()“(ignorují gradientové barevné přechody). LCP tak počítá jen samostatný box model čili velikost samotného elementu, ale ignoruje velikosti marginu, paddingu a borderu. Pokud se však následný element mění velikost za pomocí JS, LCP ho nezohlední. Ideální hodnoty u LCP jsou uvedeny v tabulce 3. V nástrojích PSI a Lighthouse se tato hodnota propisuje do celkového skóre s váhou 25 %. (52)

Hodnota LCP	Mobilní Zařízení	Desktop
Dobrá	≤ 2.5 sekundy	≤ 1.2 sekundy
Vyžaduje Zlepšení	≤ 4.0 sekundy	≤ 2.4 sekundy
Špatná	> 4.0 sekundy	> 2.4 sekundy

Tabulka 3 – Ideální hodnoty LCP (52)

3.11.1.6 Load (TTI)

Indikuje kompletní načtení HTML dokumentu, společně s CSS, JS obrázky a <iframes>. Stránka tedy může být už nějakou dobu konzumována uživatelem, avšak na pozadí se stahuje zbytek souborů jako jsou například velké obrázky. Hodnota této metriky je velice ovlivněna velikostí jednotlivých webových stánek. (53)

3.11.2 Vývojářské nástroje

Pro účely této práce se budou využívat převážně vývojářské nástroje implementované ve webových prohlížečích. Tyto nástroje nabízí funkce pro rozsáhlé a podrobné měření, mimo jiné je možná editace a interakce s jednotlivými elementy HTML a editace veškerého CSS, které se na stránce zobrazuje. (54)

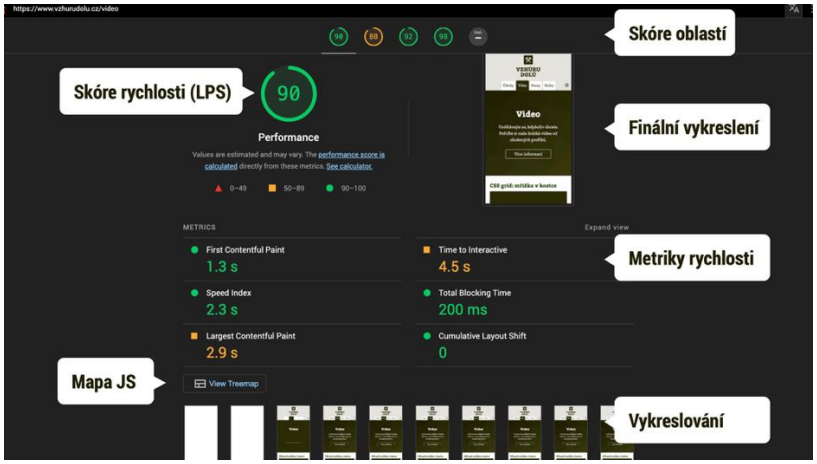
3.11.2.1 Prohlížečové Developerské Nástroje

Jsou dostupné pouze ve webových prohlížečích, nejlepší na trhu však jsou zabudované ve Firefox a Chromu. Výběr nástroje tedy závisí převážně na vývojáři a jeho osobních preferencích. V praxi se však vývojáři tyto vývojářské nástroje kombinují.

Otevírají vývojářům novou řadu možností, jak analyzovat webovou stránku. Jsou zde vidět takřka všechny informace, které prohlížeč o stránce zaznamenává. Zobrazuje veškeré HTML a CSS které je vykreslené pomocí DOM a následně nám dává možnost tyto části kódu ty také editovat. Sledovat průběh CSS animací, možnost změnit pohled na webovou stránku například z mobilního zařízení a zobrazuje JS konzoli. Pro účely měření je však nejužitečnější sekce network, která zobrazuje, jak se stránka načítá, elementy blokující toto načítání a také si lze nastavit limitaci internetového připojení. Lze tak měřit rychlosti načítání za specifických rychlostech internetového připojení. (54) (55)

3.11.2.2 Google Chrome Lighthouse

Pomocí tohoto nástroje Google radí vývojářům, jak zlepšit jednotlivé části stránky nejen z pohledu SEO, ale také úroveň designu, front-end kódu a přístupnost pro uživatele. Zjišťuje tak hodnoty jednotlivých metrik a následně je tak schopný analyzovat celkovou technickou kvalitu webu. (56)



Obrázek 31 – Ukázka Google Lighthouse (56)

3.11.3 Analyzační webové stránky

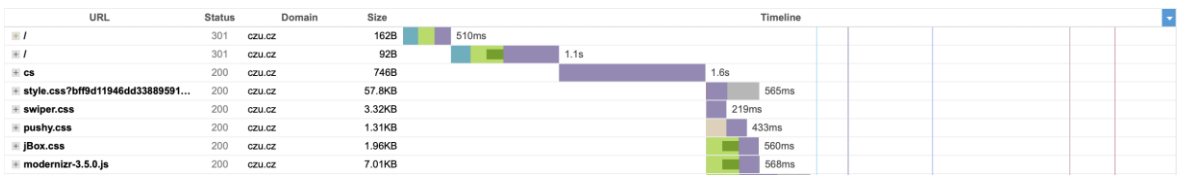
Načítání webové stránky se mimo jiné dá měřit také pomocí externích stránek, které jsou schopny změřit rychlost načítání stránky, špatnou optimalizaci obrázků a SEO společně s dalšími metrikami zmíněnými v sekci 3.11.1(Metriky pro měření). Těchto stránek existuje mnoho, avšak pro účely této práce budou níže zmíněné dostačující. (57)

3.11.3.1 Google PageSpeed Insights (PSI)

Webová stránka vyvíjena společností Google. Jedná se o nástroj, který je schopný sdělit a informovat o celkové uživatelské zkušenosti návštěvníka, který navštěvuje stránku jak z počítače, tak z mobilního zařízení. Následně vyhodnotí výsledky vlastního měření a nabízí návrhy, jak by se mohla tato uživatelská zkušenost zlepšit a sdělí celkové bodové zhodnocení webové stránky, kdy 0 je nejhorší a 100 nejlepší. (57)

3.11.3.2 GTMetrix

Jedná se o velice podobnou stránku jako PSI, avšak hlavní výhodou GTMetrixu je, že po registraci si lze navolit testovací zemi a specifické zařízení. Tato stránka nabízí jak placené, tak neplacené služby. Mimo jiné je GTMetrix schopný zobrazit podobnou detailní datovou strukturu, kterou lze zobrazit také v developerských nástrojích v prohlížeči. (58)



Obrázek 32 – Ukázka zobrazení dat na stránce GTMetrix (58)

3.12 Optimalizace načítání webové stránky

Načítání webové stránky lze také optimalizovat nikoli jen pomocí zvolení správné metody psaní CSS a počtu CSS souborů, ale i pomocí dalších způsobů, které jsou zmíněny níže v této sekci.

3.12.1 GZIP

GZIP je způsob bezztrátové komprese, která pomáhá pro zrychlení načítání webové stránky. Pokud server dostane žádost na zobrazení webové stránky, překontroluje, zda uživatelův prohlížeč podporuje gzip. V momentě, kdy ano, server bezztrátově kompresuje data a odesílá je tak koncovému uživateli, jehož prohlížeč tato data dekompresuje a zobrazí uživateli. Zkrátí se tak doba stahování a čekání na odpověď serveru. Vývojář má možnost určit velikost komprese, která bude použita. (59)

3.12.2 Minification

Minifikace je proces odstraňování a redukování přebytečných dat bez narušení plynulosti a funkčnosti kódu. Příkladem jsou komentáře, formátování a odstranění mrtvého kódu. Doporučuje se minifikovat hlavně HTML, CSS a JS soubory. Dosahuje se tak tedy až o 60% menší velikosti souborů, oproti jejich původním. Níže na obrázku 33 a 34 lze vidět rozdíl klasického zápisu CSS a jeho minimalizovanou verzi. (60)

```css
.button {
    background: white;

}

.button-2 {
    background: black;
}
```

Obrázek 33 – CSS soubor před minifikací (60)

```css
.button{background: #fff}.button-2{background: #000}
```

Obrázek 34 – CSS soubor po minifikaci (60)

3.12.3 Obrázky

Pokud není obrázek na stránce dostatečně optimalizovaný, může značně zpomalit načítání webové stránky. Vždy je třeba přemýšlet nad tím, kde bude obrázek na stránce umístěn. Pokud má sloužit pouze jako profilový obrázek, není ho tak potřeba nahrávat a načítat ve velkém rozlišení. (61)

Využití obrázků ve formátu „.WebP“ vzhledem k tomu, že mají v průměru o 25-35 % menší velikost oproti jejich původnímu formátu (.png, .jpg). Tuto formu optimalizace stránky využívá například YouTube, který od zavedení zvedl rychlost načítání stránky o 10 %. Oproti tomu Facebook hlásí až 35 % změnu u „.jpeg“ obrázků a až 80 % u „.png“. WebP nabízí bezztrátovou i ztrátovou kompresi. Kdy při bezztrátové nedochází ke zmenšení velikosti souboru. Na rozdíl od ztrátové, kdy dochází ke zmenšení velikosti souboru na úkor menší kvality. (62)

Záměna animovaných „.gif“ obrázků za video může vést k výraznému zrychlení načítání stránky. Při konvertování gifu na WebM formát lze dosáhnout stejné funkcionality jako má gif, avšak zlepší se tak doba načítání stránky. (63)

Využití scalable vector graphics (svg) je jednou ze základních metod, kdy lze nahradit například logo a ikony části svg kódu, místo klasických obrázků ve formátu „.png“ a zároveň tak pomůže SEO a celkovému hodnocení stránky. (64)

3.12.4 Cache

Když už byla data jako je JS, CSS a HTML jednou načtena, není již potřeba tento proces opakovat znovu. Prohlížeč si uloží tyto data do své paměti a při příští návštěvě webové stránky, se zásadně urychlí rychlost načtení. Cache jsou pravidla, které určuje developer, jako například jak dlouho mají být data uložena v prohlížeči. (65)

3.13 Obdobná řešení

CSS animace mohou být náročné na hardware, na webové stránce web.dev se tímto problémem zabývali, kde bylo zmíněno, několik náročných CSS vlastností, jako je „transform“ a „opacity.“ Společně s možnými řešeními, za použití vlastnosti „will-change.“ Zde se však zkoumala čistě jen HW náročnost jednotlivých vlastností animací pomocí developerských nástrojů v prohlížečích Mozilla Firefox a Google Chrome. Zde taky platí

předpoklad, že je animace zavolána za určitých okolností, nikoli při načítání webové stránky. (45)

Problematika načítání webových stránek podle počtu CSS soborů byla v určitém ohledu probírána v jedné diplomové práci psané Jiřím Kaštánkem na téma „Vliv CSS na výkon a rychlost načítání webových stránek“. U této měřené webové stránky však bylo využito technologie PHP, přesněji funkce „include“, za jejíž pomoci tak byly importovány komponenty jako jen navbar, head a footer. Bylo tak nutné, aby se využil server-side rendering. Rozsah webové stránky byl také rozlišný, kdy autor diplomové práce zvolil webovou stránku o více podstránkách. Autor diplomové práce zjistil, že se pro tento typ webové prezentace vyplatí zvolit metodu jednoho CSS soboru, kdy při volbě komponentového CSS dosahoval vyšších načítacích časů než za využití jednoho obsáhlého souboru. (66)

4 Vlastní práce

Teoretická část této práce se zabývá vlivem vlastností a nástrojů CSS na načítání webových stránek. Měření bude zkoumat náročnost na uživatelův hardware a rychlost načítání experimentální webové stránky, za určitých podmínek a při využití vybraných vlastností CSS, jako jsou například CSS Animace a počet CSS souborů. Boudou sledovány metriky, které byly zmíněny v teoretické části viz. kapitola 3.11 (Měření načítání webové stránky)

4.1 Podmínky pro měření

Každé měření vždy bude probíhat za stejných podmínek, které jsou zmíněny v kapitolách níže, v případě, že by nebylo možné těchto podmínek dosáhnout, měření nebude provedeno. Rychlost internetového připojení pro měření byla 50Mb/s pro rychlost stahování a 10Mb/s pro rychlost nahrávání. Zásadním předpokladem také je, že je webová stránka napsána dle sémantických pravidel pro psaní HTML a CSS.

4.1.1 Postup měření

Každé měření bude provedeno pětkrát, v případě, že by se naměřené hodnoty značně lišily od průměrných hodnot, toto měření bude zahozeno a nebude zaznamenáno, toto měření se bude opakovat. Naměřené hodnoty pro každou metriku budou následně zprůměrovány a přepracovány také do grafické podoby.

4.1.2 Nástroje použité pro měření

Pro měření bylo využito dvou měřících nástrojů, prvním z nich byl Page Speed Insights od společnosti Google, který je dostupný jak online, tak ve webovém prohlížeči Google Chrome. Pro ujištění bylo vyzkoušeno, zda se se výsledky měření online nástrojem liší od toho zabudovaného v prohlížeči, výsledky se nelišily. Druhým byl nástroj „performance," který se je také zabudovaný ve vývojářských nástrojích prohlížeče Google Chrome. Veškeré měření bude probíhat ve webovém prohlížeči Google Chrome. Pro hostingovou službu byla vybrána služba GitHub Pages. Tato služba je zcela zdarma, avšak je částečně omezená, ale tato omezení nebudou mít vliv na výsledky měření.

4.1.3 Zařízení využité k měření náročnosti na hardware

Pro měření bude využito desktopové zařízení iMac 2021 od společnosti Apple, v základní konfiguraci. Jeho hardwarové a softwarové prvky jsou:

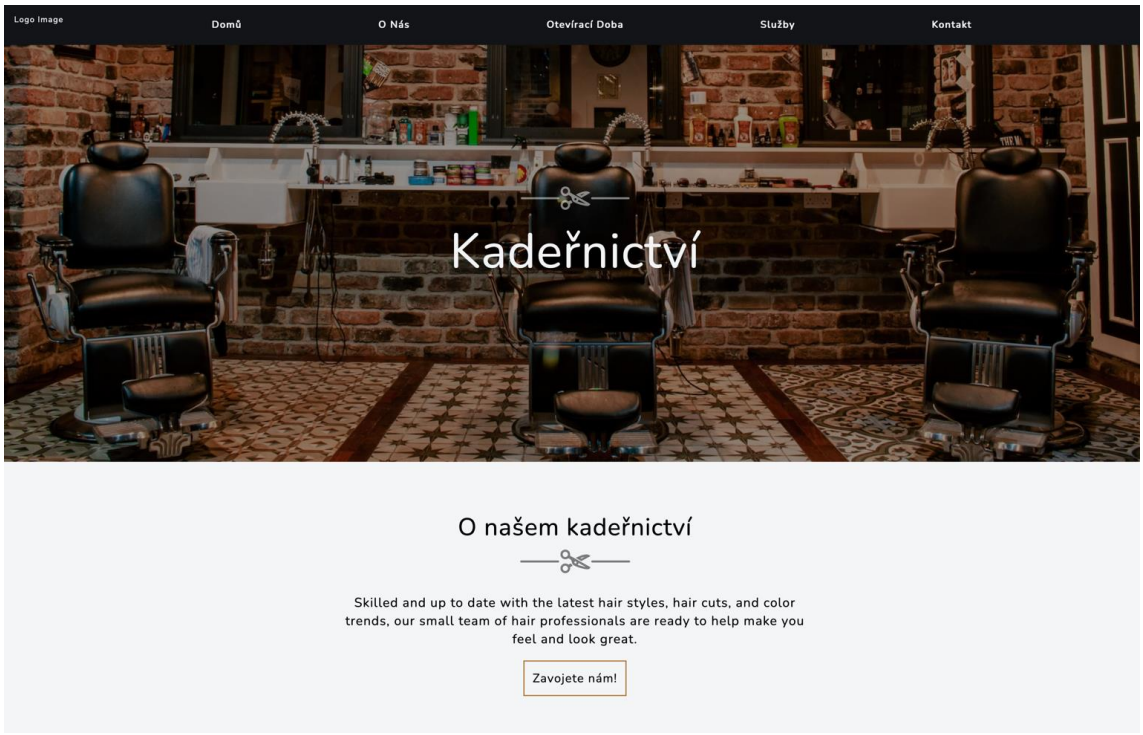
- Procesor: Apple M1, 8 jader
- Grafický čip: Integrovaný, 7 jader
- Operační paměť: 8 GB
- Typ úložiště: SSD
- Display: 24palcový 4,5k retina display
- Operační Systém: macOS Ventura verze 13.1

K diversifikaci a kontrole měření následně bude využit stolní počítač, s následujícími specifikacemi:

- Procesor: AMD Ryzen 5 3600X
- Grafická Karta: Dedikovaná – Radeon RX 5600 XT Phantom Gaming D3 6G OC
- Operační paměť: 16 GB
- Typ úložiště: SSD
- Display: 24palcový 1920x1080 fullHD
- Operační Systém: Windows 11

4.2 Experimentální webová stránka

Pro tuto práci byla použita autorem vytvořená webová stránka, která je již používána pro účely webové prezentace malého byznysu. Pro ochranu obou stran však tato stránka byla předělána na fiktivní byznys, byl změněn název, obrázky byly nahrazeny za ilustrační a veškeré texty byly přepsány. Základní zdrojový kód byl následně upravován pro účely měření.



Obrázek 35 – Výstřižek ze vzorové webové stránky.

4.2.1 Rozložení stránky

Tématem vzorové stránky je kadeřnický salón. Obsah je formulován tak, aby uživateli stručně a jasně sdělil nejzákladnější informace o salónu. Jedná se o jednostránkový web, který je plně responsivní, na všechna zařízení, od mobilního, přes tablet až na desktopové zařízení.

Stránka tedy obsahuje pouze nejzákladnější sekce. Těmi jsou: navigace, intro se sloganem, základní formace, otevírací hodiny, přednosti kadeřnictví, ceník, sekci s kontakty na jednotlivé kadeřnice a footer(patičku).

Pro účely měření bude však sekce ceník duplikována a celkově se tak na stránce bude nacházet sedmdesát sekcí, které dohromady v HTML souboru zabírají zhruba 5850 řádků,

pokud budou opomenuty řádky v hlavičce a import jednotlivých scriptů. Celkové množství CSS řádků hlavního souboru bez animací bylo 596.

4.2.2 Měření experimentální webové stránky

Při tvorbě webové stránky byly využito pouze HTML a základního JS, aby nedošlo k znehodnocení výsledků měření jiným vlivem než CSS. Značkovacím jazykem HTML byla vytvořena kostra, CSS bylo psáno bez jakéhokoliv využití frameworků, knihoven a preprocesorů. JavaScriptu bylo využito jen v případě, že by řešení pomocí CSS bylo nadměrně komplikované.

Z počátku měla webová stránka 2 CSS soubory. Veškerá stylizace byla napsána v jednom CSS souboru, v druhém se nacházely animace, které byly pro první měření z webu odstraněny. Výsledky načítání (viz. tabulka 4) byly změřeny pomocí nástroje Lighthouse v prohlížečí Google Chrome.

Metrika	Výsledky měření v sekundách					
	M1	M2	M3	M4	M5	Průměr
First Contentful Paint	0,5	0,6	0,5	0,5	0,5	0,52
Time to Interactive	0,8	0,8	0,7	0,8	1,1	0,84
Speed Index	0,5	0,6	0,5	0,5	0,5	0,52
Largest Contentful Paint	1,2	1,2	1,1	1,2	1,3	1,20

Tabulka 4 – Výsledky měření načítání stránky bez animací.

Pozn.: Vlastní zpracování naměřených dat

Bodové hodnocení nástroje PageSpeedInsight se pohybovalo v rozmezí od 96 do 99, což je opravdu dobré skóre, vzhledem k faktu, že 100 bodů je maximum. Lze pozorovat, že metrika FCP čili první zobrazení obsahu na stránce byla vždy změřena v zelených číslech, což je hodnota menší, nebo rovna 0,9 viz. kapitola 3.11.1 (Metriky pro měření). Hodnota LCP, první vykreslení největšího prvku na stránce, se však pohybovala u horní hranice, kdy hodnota 1,2 je maximální hranice pro dobré hodnocení této metriky. Pokud by tedy měla být stránka hodnocena pouze z pohledu optimalizace, bylo by možné zapracovat na zlepšení metriky LCP. V praxi se však tato metrika málokdy dostane do zelených hodnot, u obsáhlejších webových stránek.

Následné měření však zkoumá náročnost webové stránky na hardware na straně klienta. Měření bylo zaměřeno na délku běhu procesoru (CPU) a také grafického čipu (GPU). U CPU se však nachází dvě metriky, kdy „rendering“ udává, jak dlouho procesor

zpracovával zasazení jednotlivých HTML prvků do DOM a následné vypočítání velikosti a usazení na správnou pozici na stránce. Paint je však vykreslení pixelů pro již „vyrenderované“ prvky. Výsledky měření byly zaznamenány do tabulek Tabulka 5(iMac) a 6 (Stolní Počítač).

Zařízení: iMac 2021						
Metrika – Runtime	Výsledky měření v milisekundách					
	M1	M2	M3	M4	M5	Průměr
GPU	58	57	52	61	55	56,6
CPU – Rendering	60	59	96	85	57	71,4
CPU – Painting	3	3	3	3	3	3

Tabulka 5 – Výsledky měření hardwarového zatížení na zařízení iMac 2021

Pozn.: Vlastní zpracování naměřených dat

Zařízení: Stolní počítač						
Metrika – Runtime	Výsledky měření v milisekundách					
	M1	M2	M3	M4	M5	Průměr
GPU	108	99	120	77	88	98.4
CPU – Rendering	70	75	77	78	77	75.4
CPU – Painting	4	5	5	6	3	4.6

Tabulka 6 – Výsledky měření hardwarového zatížení na stolním počítači

Pozn.: Vlastní zpracování naměřených dat

Naměřené hodnoty zobrazují hardwarové zatížení v milisekundách. Hodnoty byly získány pomocí nástroje „performance“ ve webovém prohlížeči Google Chrome. Lze zjistit, že naměřené hodnoty GPU a Rendering byly proměnlivé, však hodnota Painting zůstala vždy stejná na zařízení iMac, avšak na stolním počítači byla také proměnlivá.

Obrázek 36 – Průběh načítání testovací stránky

4.3 Vliv CSS animací na načítání webové stránky

V případě že je na jedné webové stránce použito nadměrné množství animací, může jí to zpomalit. Každá jedná animace přidává prohlížeči náročnost pro zpracování, kdy prohlížeč využívá procesor a operační paměť, které při velkém množství animací můžou být přetíženy. V případě složitějších animací, jako jsou například 3D transformace. Mimo jiné mohou stránku zatížit také animace, které nastávají příliš často, jako je například „onHover“

(přejetí cursorem) efekt, nebo také animace, které běží v nekonečné smyčce. Vždy však nemají zásadní vliv na prvopočáteční načtení stránky, avšak mají vliv na plynulý průchod webovou stránkou. Hardwarová zařízení se slabšími specifikacemi se tak mohou potýkat s problémy.

4.3.1 Animace při využití vlastnosti „opacity“

Při přidání animací za využití vlastnosti „opacity“ na element <section> výsledky měření lze vidět na obrázku 37. Při tomto měření však častěji docházelo k chybnému měření, kdy níže měřené metriky měly až trojnásobnou hodnotu, tyto hodnoty byly naměřeny zhruba u jednoho z deseti případů. Výsledky měření lze vidět v tabulce 7.

```
@keyframes opacityChange {
    0% {
        opacity: 0% ;
    }

    25% {
        opacity: 25%;
    }

    50% {
        opacity:50%
    }

    75% {
        opacity: 75%;
    }

    100% {
        opacity: 100%;
    }
}

section{
    animation-name: opacityChange;
    animation-duration: 10s;
}
```

Obrázek 37 – Přidělená animace opacityChange pro section

Metrika	Výsledky měření v sekundách					
	M1	M2	M3	M4	M5	Průměr
First Contentful Paint	0,6	1,0	0,6	0,6	0,6	0,68
Time to Interactive	0,9	1,0	0,9	0,9	0,9	0,92
Speed Index	1,2	1,2	1,2	1,2	1,2	1,2
Largest Contentful Paint	1,3	1,4	1,4	1,3	1,3	1,34

Tabulka 7 – Výsledky měření stránky s použitými animacemi s vlastností „opacity“

Pozn.: Vlastní zpracování naměřených dat

V momentě, kdy každou sekci na stránce byla přidána animace na změnu opacity, hodnoty měřených metrik se změnily k horšímu, FCP a TTI se oproti stránce bez animací v průměru zhoršily o pár desítek milisekund, Avšak dochází zde k rapidnímu zhoršení SI, který se v průměru zvedl o 0,68s a LCP by vyžadovalo zlepšení, viz kapitola 3.11.1 (Metriky měření).

Následně bylo změřeno hardwarové zatížení, při načítání webové stránky, kdy na tabulce 8 lze pozorovat, že procesor zpracovával rendering v průměru déle než v případě, že na stránce nebyly použity animace. Hodnota painting se však zvýšila v průměru zhruba čtyřikrát, lze však pozorovat, že tato hodnota není přímo závislá na hodnotě rendering. S rostoucím časem renderingu roste také čas painting, neexistuje však mezi těmito hodnotami přímá úměra.

Zařízení: iMac						
Metrika – Runtime	Výsledky měření v milisekundách					
	M1	M2	M3	M4	M5	Průměr
GPU	58	51	70	59	60	59,6
CPU – Rendering	179	172	198	180	178	181,4
CPU – Painting	14	14	11	11	10	12

Tabulka 8 – Výsledky měření hardwarového zatížení na zařízení iMac 2021 na stránce s animacemi za využití vlastnosti „opacity“

Pozn.: Vlastní zpracování naměřených dat

Obrázek 38 – Průběh načítání testovací stránku s animací opacityChange

4.3.2 Animace při využití vlastnosti „transform“ a „translate“

Pro měření této vlastnosti byla využit a CSS vlastnost „translateY“ (viz. obr. 39) vzhledem k faktu, kdy při měření s vlastností „translateX“ docházelo ke chybě měření, kdy se analyzačnímu nástroji vůbec nepovedlo zachytit hodnotu LCP a vypsala „error.“ Při nahrazení této vlastnosti se tyto problémy již nevyskytovali. Pro měření byly zachovány stejné podmínky, jako pro měření animací s vlastností „opacity.“

```
@keyframes translateY {
    0% {
        transform: translateY(-10%);
    }

    25% {
        transform: translateY(10%);
    }

    50% {
        transform: translateY(-10%);
    }

    75% {
        transform: translateY(10%);
    }

    100% {
        transform: translateY(0%);
    }
}
```

```
section{
    animation-name: translateY;
    animation-duration: 10s;
}
```

Obrázek 39 – Využití animace s vlastností translateY

Po provedení měření vlivu animací za využití vlastnosti translateY bylo zjištěno, že naměřené hodnoty jsou značně vyšší, než by měla být jejich maximální doporučená hodnota pro dobré skóre. Pokud se v jeden moment použije velké množství těchto animací, může se stát, že rapidně ovlivní celkové skóre stránky. V případě, kdy tyto animace budou zavolány později, za určitých okolností, skóre se výrazně zlepší.

Metrika	Výsledky měření v sekundách					
	M1	M2	M3	M4	M5	Průměr
First Contentful Paint	1,6	1,4	2,1	1,5	1,2	1,56
Time to Interactive	1,6	1,4	2,1	1,5	1,2	1,56
Speed Index	1,6	1,8	2,1	1,8	2,4	1,94
Largest Contentful Paint	3,1	3,0	3,5	3,1	1,2	2,78

Tabulka 9 – Výsledky měření za využití animace s vlastností translateY

Pozn.: Vlastní zpracování naměřených dat

Výsledky naměřených hodnot (viz. tabulka 10) pro hardwarovou náročnost animací opět potvrzují, že zátěž procesoru roste v závislosti na počtu animovaných prvků, společně s komplexností animace a použitými vlastnostmi. Zatížení GPU se nijak v zásadě nemění.

Zařízení: iMac						
Metrika – Runtime	Výsledky měření v milisekundách					
	M1	M2	M3	M4	M5	Průměr
GPU	61	54	45	47	44	50,2
CPU – Rendering	212	226	387	235	394	290,8
CPU – Painting	16	19	23	16	22	19,2

Tabulka 10 – Výsledky měření stránky na zažízení iMac2021 s použitými animacemi s vlastností „translateY“

Pozn.: Vlastní zpracování naměřených dat

Obrázek 40 - Průběh načítání webové stránky s animacemi za použití vlastnosti translateY

V tabulce číslo 11 jsou uvedeny hodnoty, které byly získány z měření, kdy se animace stane pouze za určité situace, v tomto případě byla využita pseudotřída „: hover.“ Lze pozorovat, že u čtyř z pěti případů byly hodnoty měření naprosto totožné.

Metrika	Výsledky měření v sekundách					
	M1	M2	M3	M4	M5	Průměr
First Contentful Paint	0,7	0,6	0,6	0,6	0,6	0,62
Time to Interactive	1,0	0,9	0,9	0,9	0,9	0,92
Speed Index	0,7	0,6	0,6	0,6	0,6	0,62
Largest Contentful Paint	1,3	1,3	1,3	1,3	1,3	1,30

Tabulka 11 – Výsledky měření za využití animace s vlastností translateY, s pseudotřídou ":hover"

Pozn.: Vlastní zpracování naměřených dat

4.4 Srovnání rychlosti načítání dle počtu CSS souborů

Tato část práce se bude zaobírat měřením hodnot pro načítání webové stránky, v závislosti na počtu CSS soboru, je důležité podotknout, že práce je měřena neobsáhlé webové stránce. Čili výsledky měření budou odpovídat tomuto rozsahu webové stránky. V momentě, kdy stránka rosta na komplexnosti, hodnoty měření se mohou lišit. Pro účely tohoto měření nebudou použity žádné animace, aby nedošlo k znehodnocení výsledků.

4.4.1 Jeden CSS soubor

Tyto výsledky byly změřeny již v úvodu sekce praktické části této práce, viz. 4.2.2 (Měření experimentální webové stránky). Byly naměřeny následující hodnoty, viz. tabulka 12.

Metrika	Výsledky měření v sekundách					
	M1	M2	M3	M4	M5	Průměr
First Contentful Paint	0,5	0,6	0,5	0,5	0,5	0,52
Time to Interactive	0,8	0,8	0,7	0,8	1,1	0,84
Speed Index	0,5	0,6	0,5	0,5	0,5	0,52
Largest Contentful Paint	1,2	1,2	1,1	1,2	1,3	1,20

Tabulka 12 – Výsledky měření rychlost načítání webové stránky s jedním CSS souborem

Pozn.: Vlastní zpracování naměřených dat

4.4.2 Více CSS souborů

Pro účely tohoto měření bude využita jedna z metod zápisu CSS zvaná komponentové CSS, viz. kapitola 3.9.8 (Více CSS souborů). V tomto případě byl jeden hlavní CSS soubor, který obsahoval veškerou stylizaci rozdělen na jednotlivé CSS komponenty. Bude zde však zkoumáno, zdali má nějaký vliv na načítání stránky implementace jednotlivých CSS souborů, kdy jednou z metod je načítání souborů přímo z HTML a druhou kdy jsou CSS soubory importovány do hlavního CSS souboru, který je spojen s HTML viz obrázek 41.

Propojení potřebných CSS stylů pomocí CSS lze vidět na obrázku 41, využije se funkce @import url(název.css), takto lze importovat neomezené množství CSS souborů. Výsledky měření v tomto způsobu mají následující hodnoty. Průměr každé jedné z naměřených metrik je vyšší, než když byla veškerá stylizace umístěna v jednom souboru viz. tabulka 13.

```
@import url(navigation.css);
@import url(about.css);
@import url(openingHours.css);
@import url(services.css);
@import url(barbers.css);
@import url(contacs.css);
@import url(footer.css);
```

Obrázek 41 – Připojení CSS souborů pomocí funkce @import

Metrika	Výsledky měření v sekundách					
	M1	M2	M3	M4	M5	Průměr
First Contentful Paint	0,5	0,7	0,7	0,7	0,7	0,72
Time to Interactive	0,8	0,9	0,9	0,9	0,9	0,96
Speed Index	0,5	0,7	0,7	0,7	0,7	0,72
Largest Contentful Paint	1,2	1,4	1,4	1,7	1,3	1,42

Tabulka 13 - Výsledky měření při více CSS souborech za použití funkce @import

Pozn.: Vlastní zpracování naměřených dat

Propojení všech CSS souborů pomocí HTML viz. obrázek 42, a je umístěna do tagu <head>. Oproti implementaci pomocí @import dosáhlo toto propojení lepšího času, pro hodnotu LCP, jak lze vidět v tabulce číslo 14, avšak rozdíly všech hodnot jsou zanedbatelné.

```
<link rel="stylesheet" href="css/main.css" />
<link rel="stylesheet" href="css/about.css">
<link rel="stylesheet" href="css/contacts.css">
<link rel="stylesheet" href="css/navigation.css">
<link rel="stylesheet" href="css/services.css">
<link rel="stylesheet" href="css/openingHours.css">
```

Obrázek 42 – Připojení CSS souborů pomocí HTML

Metrika	Výsledky měření v sekundách					
	M1	M2	M3	M4	M5	Průměr
First Contentful Paint	0,7	1,1	0,7	0,7	0,7	0,78
Time to Interactive	1,0	1,1	0,9	0,9	0,9	0,96
Speed Index	0,7	1,1	0,7	0,7	0,7	0,78
Largest Contentful Paint	1,4	1,4	1,4	1,4	1,3	1,38

Tabulka 14 – Výsledky měření při více CSS souborech při propojení souborů pomocí CSS

Pozn.: Vlastní zpracování naměřených dat

5 Výsledky a diskuse

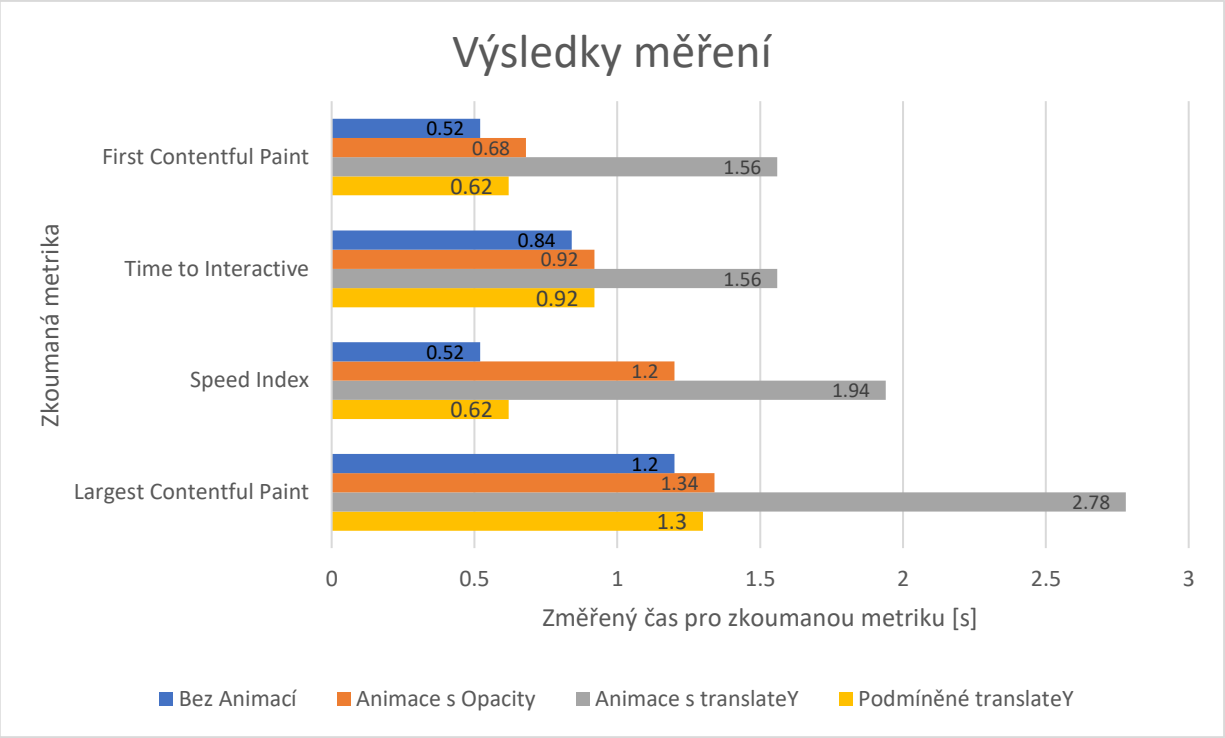
Veškeré grafické zpracování je tvořeno z dat vycházejících z měření na zařízení iMac 2021, pro každé měření však bylo provedeno také kontrolní měření, na jiném zařízení, viz. kapitola 4.1.3, kde se jednotlivé hodnoty nikdy zásadně nelišily od těch naměřených na iMacu.

5.1 Vliv CSS animací na načítání webových stránek

Velká část praktické části této práce byla zaměřena na vliv animací na načítání webových stánek, při měření bylo zkoumáno, jak se stránka načítá v momentě, kdy je na stránku přidáno větší množství animací s náročnějšími CSS vlastnostmi.

Byla provedena tři měření, kdy třetí (animace s translateY) bylo měřeno za dvou různých podmínek. Z výsledků měření vyplývá, že v případě, kdy na stránce nejsou žádné animace, její hodnocení dosahuje lepších výsledků. Animace však mohou výrazně pomoci k lepší uživatelské zkušenosti, (UX) a tím pádem tak k lepším prodejům a nižším bounce rates, viz kapitola 3.10(CSS Animace).

V momentě, kdy na stránku je přidáno větší množství animací, rychlost načítání se odvíjí od použité CSS vlastnosti pro danou animaci, jejich množství, a za jakých podmínek jsou použity. Graf 1: na ose Y lze vidět jednotlivé měřené metriky, a osa X pak značí naměřený čas v sekundách.



Graf 1 – Grafické zpracování výsledku měření načítání webové stránky za využití animací.

Pozn.: Vlastní grafické *zpracování naměřených dat*

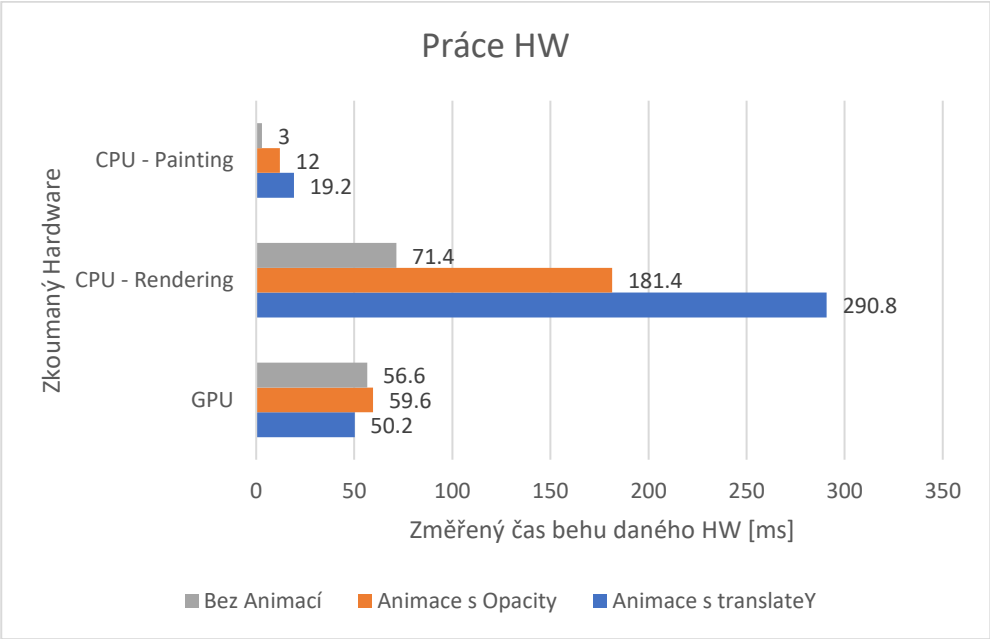
Na „Graf 1“ tak lze pozorovat, že v momentě, kdy bylo na stránku přidáno větší množství animací za použití vlastnosti „opacity,“ základní metriky byly naměřeny s vyšší hodnotou. Tři ze čtyř metrik byly ovlivněny jen o několik desítek milisekund, rapidně se však zhoršila hodnota Speed Indexu. Za použití animací s vlastností translateY se však výsledky měření rapidně mění od hodnot, které byly naměřeny bez použití jednotlivých animací. Tento problém byl vyřešen přidáním podmínky, kdy se má animace zavolat, s tímto řešením se podařilo naměřit hodnoty, které se blížili těm původním, pokud je opominut SpeedIndex. Data jsou zaznamenána v tabulce číslo 15.

Metrika [s]	Bez Animací	Animace s Opacity	Animace s translateY	Podmíněné translateY
Largest Contentful Paint	1.2	1.34	2.78	1.3
Speed Index	0.52	1.2	1.94	0.62
Time to Interactive	0.84	0.92	1.56	0.92
First Contentful Paint	0.52	0.68	1.56	0.62

Tabulka 15 – Srovnání načítání webové stránky za využití animací

Pozn.: Vlastní zpracování naměřených dat

Při měření animací byla také zkoumána náročnost na hardwarové prvky na straně uživatele. Výsledky byly úměrné předešlému měření rychlosti načítání. Pokud animace ovlivňuje rychlost načítání, ovlivňuje také náročnost na HW. Graf 2: Na ose Y lze vidět měřený HW, a osa X pak značí naměřený čas v sekundách.

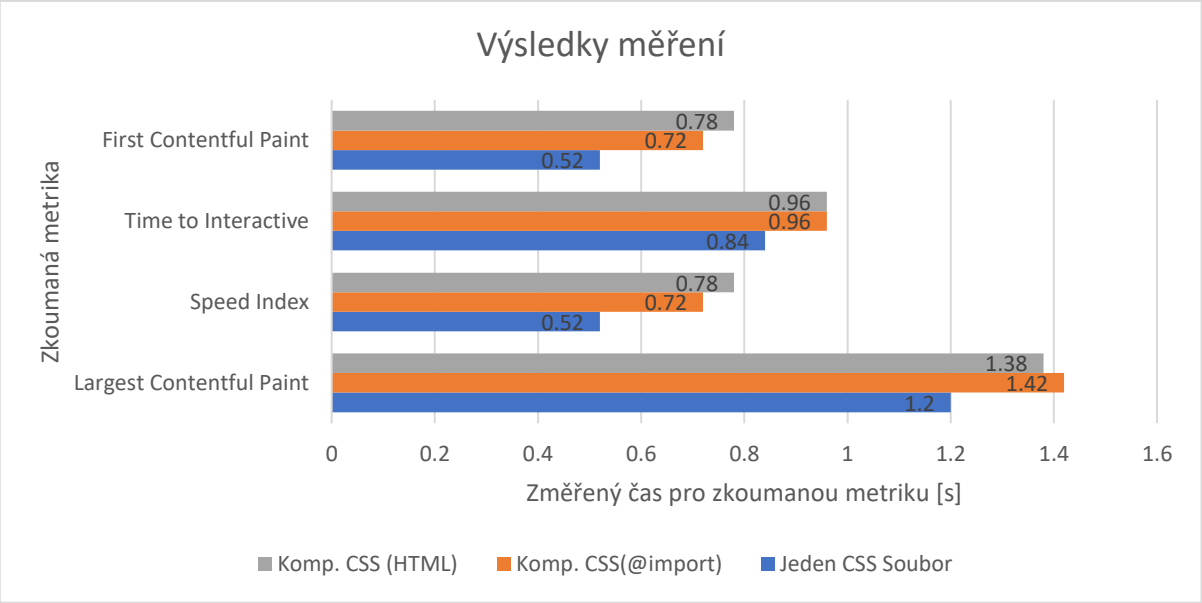


Graf 2 – Grafické zpracování výsledku měření náročnosti na hardware za využití animací.
Pozn.: Vlastní grafické *zpracování naměřených dat*

Z výsledku měření lze zjistit, že práce GPU se drží mezi 50-60 milisekundami, kdy tyto rozdíly mohou být pouze odchylkou měření, však náročnost na CPU se opět odvíjí od použité vlastnosti pro animaci, společně jejich počtem. Náročnost na rendering, kdy procesor vypočítává, kde se mají jednotlivé prvky na stránce zobrazit tak roste a s tím také hodnota painting, kdy procesor vykresluje pixely prvkům, které umístil na stránku.

5.2 Srovnání dle počtu CSS souborů

Bylo provedeno měření na dvou stylech organizace CSS. Měření pro komponentové CSS však bylo rozděleno na dvě části, kdy každá zkoumala rozdílné spojení kaskádových stylů s HTML kódem. Jedním bylo importování stylů pomocí funkce @import a druhé, kde byly jednotlivé styly umístěny přímo do hlavičky HTML. U každého stylu bylo provedeno pět měření, které nakonec bylo zprůměrováno a z těchto hodnot také vychází následující graf. Graf 3: Na ose Y lze vidět jednotlivé měřené metriky, a osa X pak značí čas.



Graf 3 – Grafické zpracování jednotlivých způsobů propojení CSS

Pozn.: Vlastní grafické *zpracování naměřených dat*

Lze tedy zřetelně vidět, že nejlépe se pro malé projekty vyplatí nejjednodušší cesta, a tou je jeden CSS soubor, kdy lze vidět, že při měření vždy docílil nejnižších výsledků. V případě, kdy by bylo nutné použít komponentové CSS a tím pádem by byl jeden soubor rozdělen na jednotlivé komponenty, z výsledků měření vyplývá, že by měla být zvolena metoda, kdy bude HTML soubor napřímo spojený pouze s hlavním stylem, do kterého následně budou importovány jednotlivé komponenty, kde byl celkový Speed Index o několik desítek milisekund lepší společně s FCP. Tato data lze vyčíst přímo z následující tabulky, viz. tabulka 16.

Metrika [s]	Jeden CSS Soubor	Komp. CSS(@import)	Komp. CSS (HTML)
First Contentful Paint	0.52	0.72	0.78
Time to Interactive	0.84	0.96	0.96
Speed Index	0.52	0.72	0.78
Largest Contentful Paint	1.2	1.42	1.38

Tabulka 16 – Srovnání jednotlivých způsobů propojení CSS

Pozn.: Vlastní zpracování naměřených dat

5.3 Diskuse

V teoretické části probíhalo experimentální měření na webové stránce, která byla předělána pro potřeby této práce. Pro tuto stránku byly následně zkoumány hodnoty vybraných metrik za využití specifických vlastností.

Vždy byla zkoumána jedna vlastnost, aby se vyvarovalo případným chybným hodnotám, měření hardwarové náročnosti bylo prováděno na dvou zařízeních, každé mělo jiné specifikace jak hardwarové, tak softwarové. Bylo využito zařízení s MacOS Ventura a zařízení s Windows11. Přestože se autorovi povedlo nalézt zdroje, viz. kapitola 3.13 (Obdobná řešení), kde již podobné zkoumání bylo zdokumentováno, měření probíhalo za jiných podmínek, a byly zkoumány jiné metriky, na jejich výsledy tak byl brán zřetel a z jejich postupu také byla získána inspirace a zkušenosti, jakých chyb se při měření vyvarovat.

Naměřené hodnoty odpovídaly očekávání, které vzniklo z poznatků zjištěných v teoretické části, někdy však docházelo k chybnému měření, kdy se zkoumané hodnoty zásadně lišily od doposud známých průměrných hodnot, nepodařilo se však zjistit, co tyto výkyvy způsobovalo.

Pro rozšíření této práce by bylo možné, zabývat se stejným měřením jako v praktické části, avšak pro každý z dostupných prohlížečů, jako je například Safari, Firefox, Edge a Opera. Bylo by také možné zkoumat více CSS vlastností, případně by bylo možné provést experimentální měření pro stránku, na které bylo využito některých z dostupných CSS frameworků, jako je například Bootstrap.

6 Závěr

V teoretické části byly shrnuty základní principy World Wide Webu, a také jakým způsobem se po něm přenášejí data a komunikace, následně bylo charakterizováno, jak webový prohlížeč tato data zpracovává pro účely načítaní webové stránky a následně je tak schopen uživateli zobrazit požadovanou webovou stránku. Následně bylo charakterizováno, co to je HTML, a jakým způsobem je úzce spojené s kaskádovými styly.

Bylo charakterizováno, co je to CSS, jeho limitace a nevýhody a základní popis CSS layoutu i responzivity. S tím jsou spojené základní metodiky, jak by se mělo přistupovat k psaní CSS, ať už se jedná o rozdělování kódu do jednotlivých souborů, pojmenovací konvence, či výběr vhodného stylu. Společně s tím tak bylo zodpovězeno, které metodiky jsou vhodné pro dané projekty, dle rozsahu. Do kapitoly metodik také nepřímo spadají preprocesory, které byly vyjmenovány a byly shrnuty jejich hlavní výhody, jak jsou nápomocné při psaní kaskádových stylů. Teoretická část se také zabývá CSS animacemi a jejich vlivem na načítání webových stránek, kde bylo teoreticky sepsáno, jak se animace tvoří, a zároveň jejich vliv na načítání stránky.

Bylo také zmíněno, jakými dalšími způsoby lze přispět k celkové optimalizaci webové stránky, mezi tyto optimalizace lze zařadit GZip, minifikaci HTML, CSS a JS kódu, a také volby vhodné velikosti a formátu obrázků a videí.

Z poznatků teoretické části a výsledků měření tak lze zjistit, jak zajistit, aby webová stránka dosahovala lepších načítacích hodnot, a zároveň tak lepšího hodnocení podle měřících nástrojů PageSpeed Insights, který je dostupný online, či ve webovém prohlížeči Google.

Prvním dílčím cílem bylo vybrat vhodné vlastnosti a nástroje CSS pro následnou analýzu. Pro účely této práce tak byly zvoleny CSS Animace, kdy bylo využíváno vlastností opacity a tranlateY, které byly spuštěny hned ze začátku načítání stránky nebo spuštěny pouze za určitých podmínek. Nástrojem pro následnou analýzu byl zvolen počet CSS souborů, kde bylo využito jednoho CSS souboru a komponentového CSS.

Druhým dílčím cílem bylo vytvořit nebo zvolit vhodné testovací webové stránky, kdy pro účely praktické části této práce byla přetvořena webová stránka, která je již využívána pro internetovou prezentaci malého byznysu, tato stránka byla přetvořena na nový fiktivní byznys a následně na ní byla prováděna veškerá měření.

Třetím dílčím cílem bylo navrhnout a provést experimentální měření na vybrané či vytvořené experimentální stránce, měření tak probíhalo na experimentální webové stránce, vytvořenou autorem práce. Experimentální měření probíhalo na základě poznatků získaných z teoretické části.

Hlavním cílem této práce však byla problematika načítání webové stránky na straně klienta, kde bylo třeba zhodnotit vliv vybraných vlastností a nástrojů CSS na načítání webové stránky. Byly měřeny metriky, kterými se zabývala teoretická část, ze které bylo zjištěno, že animace mohou mít menší i větší dopad na načítání stránky, kdy mezi hlavní faktory patří složitost animací, jejich počet, zdali animace nastanou jen za určitých okolností a také jaká vlastnost je využita pro danou animaci. Byly porovnány výsledky měření stránky, na které se nenacházely žádné animace, na kterou byly následně přidány dva typy animací, jedna s vlastností „opacity“ a druhá s „translateY.“ Z měření vyplynulo, že animace mají negativní vliv na načítání a náročnost stránky, kdy animace s vlastností opacity dosahovala daleko lepších výsledků, nežli animace s „translateY.“ Z výsledků měření tak plyne doporučení, aby byl počet animací načítajících se ihned při načítání stránky omezen na nezbytné pro UX perspektivu a zbylé byly omezeny podmíněným spuštěním, případně se vyvarovat využití většího množství animací s vlastností „translateY“.

Při zkoumání vlivu počtu CSS souborů na načítání webové stránky bylo zjištěno, že u menších webových prezentací, jako je experimentální webová stránka, která byla využita pro zkoumání a analýzu, se vyplatí využití jednoho velkého CSS a nerozdělovat ho na jednotlivé komponenty. Při využití komponentového CSS bylo zjištěno, že se z pohledu načítacích časů vyplatí importovat jednotlivé CSS sobory přímo do hlavního CSS soboru, nikoli do hlavičky HTML souboru, který je s HTML souborem spojený.

7 Seznam použitých zdrojů

1. **W3 CONTRIBUTOR.** Help and FAQ. *W3.* [Online] [Citace: 17. Srpen 2022.] https://www.w3.org/Help/#webinternet.

2. **MOZILLA.** What is a web browser? *Mozilla.* [Online] [Citace: 17. Srpen 2022.] https://www.mozilla.org/en-US/firefox/browsers/what-is-a-browser/.

3. **NEILD, David.** Which Browser Engine Powers Your Web Browsing and Why Does It Matter? *Gizmodo.* [Online] [Citace: 17. Srpen 2022.] https://www.gizmodo.com.au/2022/08/which-browser-engine-powers-your-web-browsingand-why-does-it-matter/.

4. **W3 CONTRIBUTOR.** Tim Berners-Lee. *W3.* [Online] 20. Listopad 2022. https://www.w3.org/People/Berners-Lee/.

5. **DEVELOPER MOZILLA.** HTML: HyperText Markup Language. *MDN.* [Online] [Citace: 20. Listopad 2022.] https://developer.mozilla.org/en-US/docs/Web/HTML.

6. **DUCKETT, Jon.** *HTML & CSS: design and build websites.* místo neznámé : John Wiley & sons, inc. ISBN 1118008189.

7. **DEVELOPER MOZILLA.** CSS: Cascading Style Sheets. *MDN.* [Online] [Citace: 18. Srpen 2022.] https://developer.mozilla.org/en-US/docs/Web/CSS.

8. **GEEKSFORGEEKS CONTRIBUTOR.** Advantages and Disadvantages of CSS. *GeeksForGeeks.* [Online] 23. Prosinec 2020. [Citace: 27. Leden 2023.] https://www.geeksforgeeks.org/advantages-and-disadvantages-of-css/.

9. **MICHÁLEK, Martin.** *CSS: MODERNÍ LAYOUT.* místo neznámé : Michálek Martin – Vzhůru dolů, 2022. ISBN 978-80-88253-07-5..

10. **DEVELOPER MOZILLA.** An overview of HTTP. *MDN.* [Online] [Citace: 7. Srpen 2022.] https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview.

11. **CLOUDFLARE CONTRIBUTORS.** What is HTTPS? *CloudFlare.* [Online] [Citace: 7. Srpen 2022.] https://www.cloudflare.com/learning/ssl/what-is-https/.

12. **JAVATPOINT CONTRIBUTOR.** What is a Webpage. *Java Point.* [Online] [Citace: 17. Srpen 2022.] https://www.javatpoint.com/what-is-a-webpage.

13. **HIWARLE, Uday.** How the browser renders a web page? — DOM, CSSOM, and Rendering. *Medium.* [Online] [Citace: 6. Srpen 2022.] https://medium.com/jspoint/how-the-browser-renders-a-web-page-dom-cssom-and-rendering-df10531c9969.

14. **CLOUDFLARE CONTRIBUTOR.** What is DNS? | How DNS works. *Cloud Flare.* [Online] [Citace: 18. Srpen 2022.] https://www.cloudflare.com/learning/dns/what-is-dns/.
15. **ACADEMYBIT2ME CONTRIBUTOR.** What is and how does the DNS server work? *bit2me.* [Online] [Citace: 18. Srpen 2022.] https://academy.bit2me.com/en/what-is-dns-server/.
16. **GILLIS, Alexander S.** web server. *TechTarget.* [Online] [Citace: 20. Listopad 2022.] https://www.techtarget.com/whatis/definition/Web-server.
17. **JAVATPOINT CONTRIBUTOR.** Web Servers. *JavaTPoint.* [Online] [Citace: 20. Listopad 2022.] https://www.javatpoint.com/web-servers.
18. **DEVELOPER MOZILLA.** Populating the page: how browsers work. *MDN.* [Online] [Citace: 7. Srpen 2022.] https://developer.mozilla.org/en-US/docs/Web/Performance/How_browsers_work.
19. **MILLER, Jason a OSMANI, Addy.** Rendering on the Web. *Web Dev.* [Online] [Citace: 7. Srpen 2022.] https://web.dev/rendering-on-the-web.
20. **DEVELOPER MOZILLA.** Organizing your CSS. *Learn web development | MDN.* [Online] [Citace: 8. Srpen 2022.] https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Organizing.
21. **WEBDEV CONTRIBUTOR.** Extract critical CSS. *Web Dev.* [Online] [Citace: 8. Srpen 2022.] https://web.dev/extract-critical-css/.
22. **MICHÁLEK, Martin.** Critical CSS a zrychlení zobrazení stránky. *Vzhůru Dolů.* [Online] [Citace: 8. Srpen 2022.] https://www.vzhurudolu.cz/blog/35-critical-css.
23. —. BEM: Pojmenovávací konvence pro třídy v CSS. *Vzhůru Dolů.* [Online] [Citace: 5. Srpen 2022.] https://www.vzhurudolu.cz/prirucka/bem.
24. **GETBEM.** Introduction. *Get BEM.* [Online] [Citace: 5. Srpen 2022.] http://getbem.com/introduction/.
25. **MICHÁLEK, Martin.** OOCSS: objektové psaní CSS. *Vzhůru Dolů.* [Online] [Citace: 6. Srpen 2022.] https://www.vzhurudolu.cz/prirucka/oocss.
26. **KEYCDN.** OOCSS - The Future of Writing CSS. *KeyCdn.* [Online] [Citace: 6. Srpen 2022.] https://www.keycdn.com/blog/oocss.
27. **SNOOK, Jonathan.** *Scalable and Modular Architecture for CSS.* místo neznámé : Snook.ca Web Development, Incorporated, 2012, 2012. 0985632100.

28. **GAJIC, Slobodan.** Exploring SMACSS: Scalable and Modular Architecture for CSS. *Toptal.* [Online] [Citace: 6. Srpen 2022.] https://www.toptal.com/css/smacss-scalable-modular-architecture-css.
29. **MAHMUT, Meşe.** What is SMACSS. *Medium.* [Online] [Citace: 6. Srpen 2022.] https://medium.com/@mahmutmese86/what-is-smacss-110f678c14f.
30. **PARASHAR, Pankaj.** The Making of Atomic CSS: An Interview With Thierry Koblentz. *CSS Tricks.* [Online] [Citace: 7. Srpen 2022.] https://css-tricks.com/thierry-koblentz-atomic-css/.
31. **SAUM, Joan.** An Introduction to Atomic CSS. *Better Programing.* [Online] [Citace: 7. Srpen 2022.] https://betterprogramming.pub/an-introduction-to-atomic-css-880cb02ad57f.
32. **POLACEK, John.** Let's Define Exactly What Atomic CSS is. *CSS Tricks.* [Online] [Citace: 7. Srpen 2022.] https://css-tricks.com/lets-define-exactly-atomic-css/.
33. **ENGENEERING .** Rebuilding our tech stack for the new Facebook.com. *Engineering FB.* [Online] [Citace: 7. Srpen 2022.] https://engineering.fb.com/2020/05/08/web/facebook-redesign/.
34. **DEVELOPER MOZILLA.** CSS preprocessor. *MDN.* [Online] [Citace: 7. Srpen 2022.] https://developer.mozilla.org/en-US/docs/Glossary/CSS_preprocessor.
35. **MONUS, Anna.** Popular CSS Preprocessors With Examples: Sass, Less & Stylus. *Raygun.* [Online] [Citace: 7. Srpen 2022.] https://raygun.com/blog/css-preprocessors-examples/.
36. **EVANTO.** 10 CSS Preprocessors Worth Considering. *Evanto.* [Online] [Citace: 7. Srpen 2022.] https://www.envato.com/blog/css-preprocessors/.
37. **LESSCSS.** Using Less.js. *Less CSS.* [Online] [Citace: 7. Srpen 2022.] https://lesscss.org/usage/.
38. **ITNETWORK CONTRIBUTOR.** LESS: CSS preprocesor pro pohodlnější stylování. *IT Network.* [Online] [Citace: 7. Srpen 2022.] https://www.itnetwork.cz/html-css/css3/zdrojakoviste/less-css-preprocesor.
39. **COYER, Chris.** One, Two, or Three. *CSS-Tricks.* [Online] 14. Červen 2012. [Citace: 2022. Listopad 20.] https://css-tricks.com/one-two-three/.
40. **STACKOVERFLOW CONTRIBUTORS.** Single huge .css file vs. multiple smaller specific .css files? *Stackoverflow.* [Online] [Citace: 20. Listopad 2022.] https://stackoverflow.com/questions/2336302/single-huge-css-file-vs-multiple-smaller-specific-css-files.

41. **DEVELOPER MOZILLA.** Using CSS animations. *MDN.* [Online] [Citace: 14. Leden 2023.] https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Animations/Using_CSS_animations.
42. **NNGROUP CONTRIBUTOR.** The Role of Animation and Motion in UX. *Nielsen Norman Group logoNielsen Norman Group.* [Online] [Citace: 18. Leden 2023.] https://www.nngroup.com/articles/animation-purpose-ux/.
43. **DEVELOPER MOZILLA.** Using CSS transitions. *MDN.* [Online] [Citace: 14. Leden 2023.] https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Transitions/Using_CSS_transitions.
44. —. Using CSS animations. *MDN.* [Online] [Citace: 14. Leden 2023.] https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Animations/Using_CSS_animations.
45. **ANDREW, Rachel a BASQUES, Kayce.** How to create high-performance CSS animations. *Web Dev.* [Online] https://web.dev/animations-guide/.
46. **UXPLANET CONTRIBUTOR.** How page speed affects Web User Experience. *UXPlanet.* [Online] 29. Prosinec 2019. [Citace: 27. Listopad 2022.] https://uxplanet.org/how-page-speed-affects-web-user-experience-83b6d6b1d7d7.
47. **FREEMAN, Chris.** What does a 'Bounce Rate' mean for UX? *Pixel Tree.* [Online] 7. Září 2022. [Citace: 27. Listopad 2022.] https://pixeltreemedia.co.uk/what-does-bounce-rate-mean-for-my-ux/.
48. **MICHÁLEK, Martin.** Co je „Doba do načtení prvního bajtu" (aneb „Time To First Byte" aneb TTFB)? *Vzhůru Dolů.* [Online] 4. Duben 2019. [Citace: 27. Listopad 2022.] https://www.vzhurudolu.cz/prirucka/ttfb.
49. —. Událost DOM Content Loaded (DCL). *Vzhůru Dolů.* [Online] 2019. Duben 16. [Citace: 27. Listopad 2022.] https://www.vzhurudolu.cz/prirucka/udalost-dcl.
50. **DEVELOPER CHROME.** Speed Index. *Chrome Delevopers.* [Online] [Citace: 15. Leden 2023.] https://developer.chrome.com/en/docs/lighthouse/performance/speed-index/.
51. **MICHÁLEK, Martin.** Metrika „První vykreslení obsahu" (First Contentful Paint, FCP). *Vzhůru Dolů.* [Online] 16. Březen 2019. [Citace: 27. Listopad 2022.] https://www.vzhurudolu.cz/prirucka/metrika-fcp.
52. —. Metrika „Největší vykreslení obsahu" (Largest Contentful Paint, LCP): Kdy se vykreslí hlavní obsah stránky? *Vzhůru Dolů.* [Online] 7. Srpen 2020. [Citace: 27. Listopad 2022.] https://www.vzhurudolu.cz/prirucka/metrika-lcp.

53. —. Událost Load. *Vzhůru Dolů.* [Online] 16. Duben 2019. [Citace: 27. Listopad 2022.] https://www.vzhurudolu.cz/prirucka/load.
54. **DEVELOPERS CHROME.** Overview. *Chrome Developers.* [Online] [Citace: 27. Listopad 2022.] https://developer.chrome.com/docs/devtools/overview/.
55. **DEVELOPER MOZILLA.** Firefox DevTools User Docs. *Firefox DevTools.* [Online] [Citace: 27. Listopad 2022.] https://firefox-dev.tools.
56. **MICHÁLEK, Martin.** Lighthouse: audit webu od Google. *Vzhůru Dolů.* [Online] 16. Listopad 2021. [Citace: 27. Listopad 2022.] https://www.vzhurudolu.cz/prirucka/lighthouse.
57. **DEVELOPERS GOOGLE.** About PageSpeed Insights. *Developers Google.* [Online] [Citace: 26. Listopad 2022.] https://developers.google.com/speed/docs/insights/v5/about.
58. **GTMETRIX.** Resources. *GTMetrix.* [Online] [Citace: 26. Listopad 2022.] https://gtmetrix.com/resources.html.
59. **STACKPATH CONTRIBUTOR.** What is Gzip? *Stack Path.* [Online] [Citace: 7. Srpen 2022.] https://www.stackpath.com/edge-academy/what-is-gzip/.
60. **DEVELOPERS GOOGLE.** Minify Resources (HTML, CSS, and JavaScript). *Developers Google.* [Online] [Citace: 7. Srpen 2022.] https://developers.google.com/speed/docs/insights/MinifyResources.
61. **DUÒ, Matteo.** 9 Quick Ways to Improve Page Loading Speed. *Hub Spot.* [Online] [Citace: 7. Srpen 2022.] https://blog.hubspot.com/marketing/how-to-reduce-your-websites-page-speed.
62. **HEMPENIUS, Katie.** Use WebP images. *Web Dev.* [Online] [Citace: 7. Srpen 2022.] https://web.dev/serve-images-webp/.
63. **DJIRDEH, Houssein a WAGNER, Jeremy.** Replace animated GIFs with video for faster page loads. *Web Dev.* [Online] [Citace: 7. Srpen 2022.] https://web.dev/replace-gifs-with-videos/.
64. **ADOBE.** PNG vs. SVG. *Adobe.* [Online] [Citace: 7. Srpen 2022.] https://www.adobe.com/creativecloud/file-types/image/comparison/png-vs-svg.html.
65. **BARON, Alexis Le.** Snoweb. *5 effective rules to optimize page load time.* [Online] [Citace: 7. Srpen 2022.] https://www.snoweb.io/en/website/loading-webpage/.
66. **KAŠTÁNEK, Jiří.** Vliv CSS na výkon a rychlost načítání webových stránek. [Online] [Citace: 27. Leden 2023.]

https://is.czu.cz/auth/lide/clovek.pl?zalozka=7;id=172389;studium=236039;zp=255688;download_prace=1.

8 Seznam obrázků, tabulek, grafů a zkratek

8.1 Seznam obrázků

8.2 Seznam tabulek

8.3 Seznam grafů

8.4 Seznam použitých zkratek

CSS – Cascading Styl Sheets – Kaskádové styly.
HTML – Hypertext Markup Language – Značkovací jazyk pro tvorbu webových stránek
JS – JavaScript – Scriptovací jazyk.
XML – Extensible Markup Language – Značkovací jazyk pro strukturaci dat.
HTTP – Hypertext Transfer Protocol – Internetový protokol sloužící pro přenos hypertextových dokumentů.
HTTPS – Hypertext Transfer Protocol Secure – Zabezpečený internetový protokol sloužící pro přenos hypertextových dokumentů.
TSL – Transport Layer Security – Síťový protokol, zapezpečující bezpečnou komunikaci v síti.
SSL – Secure Socket Layer – technologie, která zabezpečuje bezpečnou client–server komunikaci.
BEM – Block, Element, Modifier – Metodika a souhrn pravidel pro psaní CSS.
OOCSS – Object Oriented CSS – Metodika a souhrn pravidel pro psaní CSS.
ACSS – Atomic CSS – Metoda a souhrn pravidel pro psaní CSS.

SASS – Syntactically Awesome Style Sheets – Scriptovací jazyk, který slouží jako preprocesor pro CSS.

LESS – Leaner Style Sheets – Programovací jazyk a preprocesor pro CSS.

SMACSS – Scalable and Modular Architecture for CSS – Metodika a souhrn pravidel pro psaní CSS.

UX – User Experience – Uživatelská zkušenost.

UI – User Interface – Uživatelské rozhraní

DOM – Document Object Model – Programovací API pro HTML a XML dokumenty.

API – Application Programming Interface – Způsob jakým spolu komunikují dva počítačové programy

XML – Extensible Markup Language – značkovací jazyk

CSSOM – CSS Object Model – soubor API, který umožňuje manipulaci s CSS

SSR – Server – Side Rendering – renderování webové stránky na straně serveru

CSR – Client – Side Rendering – renderování renderování na straně klienta

FP – First Paint – Doba, která udává čas od začátku načítání stránky, do zobrazení prvního pixelu

FCP – First Contentful Paint – Doba, která udává čas od začátku načítání stránky, do zobrazení obsahu stránky

TTI – Time to Interactive (TTI) – Doba, za kterou se od začátku načítání stane stránka interaktivní.

SI – Speed Index – Metrika pro měření načítání webové stránky

GIF – Graphic Interchange Format – Pohyblivý obrázek

WebM – Web Media – Novodový doporučený formát pro videa na webové stránce.

WebP – Web Picture – Novodobý doporučený formát pro obrázky na webové stránce.

SVG – Scalable Vector Graphic – škálovatelná vektorová grafika.

SEO – Search Engine Optimalization – Věda zabývající se optimalizací webové stránky pro lepší vyhledávání vyhledávači.

WWW – World Wide Web

URL – Uniform Resource Locator

PSI – Page Speed Insight

CPU – Central Processing Unit – Procesor

GPU – Graphic Processing Unit – Grafická karta / grafický čip

MB – Mega Byte – Velikostní jednotka, která se využívá pro určení velikosti dat

Mb/s – Mega bit za sekundu – Jednotka, která se využívá pro určení datové propustnosti / rychlostni intenetového připojení

GB – Giga Byte – Velikostní jednotka, která se využívá pro určení velikosti dat

Px – Pixel – Nejmenší bod na obrazovce