Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Vliv vybraných vlastností a nástrojů CSS na rychlost načítání webové stránky

Vladimír Turek

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Vladimír Turek

Informatika

Název práce

Vliv vybraných vlastností a nástrojů CSS na rychlost načítání webové stránky

Název anglicky

The effect of selected CSS features and tools on web page loading speed

---

Cíle práce

Cílem teoretické části této práce je před samotnou analýzou rychlosti načítání webové stránky vysvětlit a popsat tvorbu webových stránek a přiblížit znalosti v oblasti nových HTML značek, CSS nástrojů a frameworků, měření rychlosti načítání webových stránek a jejich optimalizace.
Cílem praktické části této práce je zrealizovat analýzu výkonnosti na odlišně složitých webových stránkách. Ty budou vytvořeny za pomocí různých nástrojů a frameworků CSS, které jsou vhodné pro vznik optimalizovaného webu. Následně bude provedeno zhodnocení výsledků a vyvození závěrečných doporučení, které nástroje, vlastnosti a frameworky CSS jsou rychlé, jaké mají největší vliv na rychlost načítání, které jsou přívětivé pro práci a které je tedy vhodné pro tvorbu webových stránek použít.

Metodika

Pro dosažení cílů této práce bude analyzována problematika práce s CSS nástroji a frameworky pro tvorbu webových stránek a analýzu jejich rychlosti načítání. Budou popsány potřebné teoretické znalosti v této oblasti. Dojde k vytvoření různých typů webových stránek za pomoci vhodných moderních nástrojů a frameworků CSS. Tyto webové stránky budou následně analyzovány z hlediska rychlosti jejich načítání pomocí měřícího nástroje. U vytvořených stránek se bude zkoumat vliv implementace nástrojů CSS, jak ovlivní rychlost načítání stránky daný nástroj CSS a jak bude čas ovlivněn při zpracování stránky přes framework. Bude se zkoumat, jaký vliv má použití nových vlastností a nástrojů oproti dříve používaným a také rychlost CSS při vyčištění kódu od přebytečných vlastností a hodnot. Měření budou provedena v různých prohlížečích a při různých rychlostech internetového připojení. Všechna získaná a naměřená data budou hromadně zpracována a zhodnocena. Z těchto dat budou vyvozena doporučení, která budou moci využít vývojáři webových stránek při své tvorbě a vytvořit tak optimalizované a rychlé webové stránky.

Doporučený rozsah práce

30-60 stran

Klíčová slova

webové stránky, měření rychlosti načítání webu, optimalizace webu, CSS nástroje, CSS frameworky

---

Doporučené zdroje informací

GRANT, J. Keith. CSS in Depth. Shelter Island, NY: Manning publications, 2018. ISBN 978-1617293450.

MAYER, A. Eric. CSS Pocket Reference: Visual presentation for the web. 5th edition. 2018: O'Reilly Media, Inc, USA. ISBN 9781492033394.

SHENOY, Aravind a Anirudh PRABHU. CSS Framework Alternatives. Mumbai, India: apress, 2018. ISBN 978-1484233986.

Předběžný termín obhajoby

2022/23 LS – PEF

Vedoucí práce

Ing. Dana Vynikarová, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 31. 10. 2022

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 24. 11. 2022

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 15. 03. 2024

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci „Vliv vybraných vlastností a nástrojů CSS na rychlost načítání webové stránky“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce 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 15. 3. 2024 _____

Poděkování

Rád bych touto cestou poděkoval Ing. Daně Vynikarové, Ph.D. za odborné vedení, poskytnuté rady a konzultaci při psaní práce.

Vliv vybraných vlastností a nástrojů CSS na rychlost načítání webové stránky

Abstrakt

Tato bakalářská práce se věnuje analýze vlivu vybraných vlastností a nástrojů kaskádových stylů (CSS) na rychlost načítání webových stránek. Cílem práce je identifikovat klíčové CSS vlastnosti a nástroje, které mají největší dopad na dobu načítání stránky a navrhnout optimální postupy pro vývojáře webových stránek. První část práce je zaměřena na vysvětlení základních principů CSS, představení způsobu měření a analýzy rychlosti načítání webových stránek a také metrik, které se k analýze a následnému porovnávání používají. Především je kladen důraz na detailní vysvětlení celého procesu načítání webové stránky od počátku až do konce. Práce také poukazuje na význam komplexního porozumění vztahu mezi designem, funkcionalitou a výkonností v kontextu webového vývoje. V praktické části je provedena kvantitativní analýza vybraných webových stránek za použití různých CSS stylů. Metodologie testování zahrnuje měření doby načítání stránek pomocí nástrojů Google PageSpeed Insights a Chrome DevTools. V závěrečné části jsou shrnuty hlavní poznatky práce a diskutovány možné směry pro další výzkum. Tato bakalářská práce přináší ucelený pohled na vztah mezi používáním CSS a výkonností webových stránek. Klíčovým přínosem práce je poskytnutí konkrétních doporučení pro vývoj efektivnějších a rychlejších webových stránek s využitím moderních CSS technik a postupů.

Klíčová slova: webové stránky, optimalizace webu, měření rychlosti načítání webu, Web Vitals, webový prohlížeč, vykreslování webové stránky, CSS frameworky, CSS nástroje

The effect of selected CSS features and tools on web page loading speed

Abstract

This bachelor thesis is devoted to the analysis of the influence of selected properties and tools of Cascading Style Sheets (CSS) on the loading speed of web pages. The aim of the thesis is to identify the key CSS properties and tools that have the greatest impact on page load time and to propose optimal practices for web developers. The first part of the thesis focuses on explaining the basic principles of CSS, introducing how to measure and analyze web page load speed, as well as the metrics used for analysis and subsequent comparison. Above all, the focus is on a detailed explanation of the entire process of loading a web page from start to finish. The paper also highlights the importance of a comprehensive understanding of the relationship between design, functionality and performance in the context of web development. In the practical part, a quantitative analysis of selected web pages using different CSS styles is performed. The testing methodology involves measuring page load times using Google PageSpeed Insights and Chrome DevTools. The final section summarizes the main findings of the paper and discusses possible directions for further research. This bachelor thesis provides a comprehensive view of the relationship between CSS usage and website performance. A key contribution of the thesis is the provision of specific recommendations for developing more efficient and faster websites using modern CSS techniques and practices.

Keywords: website, website optimization, website loading speed measurement, Web Vitals, web browser, website rendering, CSS frameworks, CSS tools

Obsah

1 Úvod

Internetové technologie a web design neustále zvyšují tempo ve svém vývoji a otázka rychlosti načítání webových stránek je stále zásadnější. V digitálně orientovaném světě je internetové prostředí neodmyslitelnou součástí našich životů. Okamžitý, plynulý a hlavně rychlý přístup k informacím je v současné době nejen základním požadavkem uživatele, ale také klíčovým faktorem pro úspěch jakéhokoli webového projektu. Volba tohoto tématu byla motivována mým zájmem o prohloubení znalostí v oblasti tvorby a optimalizace webu a také přesvědčením, že i malé detaily, jakými právě vlastnosti CSS jsou, mohou mít významný dopad na celkový uživatelský prožitek, ale i na další skutečnosti. Ve světě, kde se každá sekunda zpoždění může rovnat ztrátě potenciálního zákazníka, je důležité porozumět tomu, jak tyto aspekty fungují, jak ovlivňují rychlost a efektivitu webových stránek.

CSS je základním stavebním kamenem webového designu, který umožňuje tvůrcům stránek definovat vizuální prezentaci svých webových stránek. Od barvy a velikosti textu po složitější animace a responzivní design. Rychlost načítání webové stránky je v podstatě prvním kontaktem uživatele s danou stránkou a může výrazně ovlivnit jeho počáteční dojem a případný návrat. Význam sledování vlivu CSS na rychlost načítání webových stránek spočívá v tom, že poskytuje hlubší porozumění, jak i malé změny v kódu, nebo jeho struktuře, mohou mít velký dopad na celkový uživatelský prožitek a dostupnost webového obsahu. Uživatelé internetu se čím dál tím více spoléhají na bezprostřední získání informací, a je proto nezbytné, aby webové stránky byly optimalizovány pro maximální výkon. Rychlost načítání není ovšem jen otázkou uživatelského pohodlí. Je to také klíčový faktor pro SEO (Search Engine Optimization) a může tak výrazně ovlivnit viditelnost webové stránky ve vyhledávačích, což může mít zásadní vliv na jeho návštěvnost. To znamená, že webové stránky s lepší rychlostí načítání mají potenciál dosáhnout vyšších pozic ve výsledcích vyhledávání, což může vést k vyšší viditelnosti a nárůstu návštěvnosti.

Tato práce se zabývá specifickými vlastnostmi CSS, které byly vybrány na základě jejich potenciálního dopadu na rychlost načítání. Předmětem zkoumání nejsou pouze samotné vlastnosti, ale i způsoby, jakými jsou implementovány a jak jejich použití ovlivňuje celkový výkon webové stránky. Přitom je pozornost věnována tomu, jak můžeme tyto vlastnosti optimalizovat, přizpůsobit a měřit, aby bylo dosaženo lepšího výkonu bez kompromisů v oblasti designu a uživatelského prožitku. Mimo to je kladen důraz i na obecnou funkcionalitu tvorby webových stránek, vysvětlení struktury a základních principů a vztahů, od serverové

infrastruktury až po konkrétní vlastnosti webového designu, technologie a principy, které jsou klíčové k pochopení toho, jak je webová stránka načítána a jaké faktory do dění vstupují.

Úvodní část bakalářské práce se věnuje širšímu kontextu fungování internetu a webových stránek. První kapitola teoretické části práce poskytuje čtenáři ucelený přehled o World Wide Webu a o jeho základních komponentách, jako jsou URL, TCP, IP, HTTP, ISP, a DNS a objasňuje jejich roli v ekosystému internetu. Dále se práce zabývá základy a technologiemi webového rozhraní, přičemž speciální pozornost je věnována modelu klient-server. Následně se teoretická část zaměřuje na webový prohlížeč jako primární nástroj pro přístup k webovým stránkám a podrobně rozebírá proces vykreslování webových stránek. Nezbytnou součástí teoretické části je pohled na jazyk CSS, včetně jeho modulárního charakteru a konkrétních „drahých“ vlastností, které mohou významně ovlivnit výkonnost načítání. Teoretická část dále obsahuje pohled na moderní metody analýzy a měření výkonu webových stránek, včetně konceptu Web Vitals, které poskytují klíčové ukazatele pro hodnocení uživatelského prožitku, optimalizace a responzivity webových stránek. Praktická část práce je zaměřena na důkladnou analýzu, měření a testování. Práce podrobně prozkoumává, jak různé layoutové techniky CSS – float, flexbox a grid ovlivňují rychlost načítání. Výzkumná část také zjišťuje, jak se projevují „drahé“ vlastnosti CSS v praxi. V další fázi je zhodnocen dopad používání CSS frameworků na rychlost načítání webových stránek a je provedeno porovnání výkonu reprezentativní webové stránky s použitím čistého CSS oproti frameworkovému CSS. Závěrečná kapitola se věnuje specifickému vlivu způsobu připojení CSS souborů na rychlost načítání webové stránky.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem teoretické části této práce je před samotnou analýzou rychlosti načítání webové stránky vysvětlit a popsat tvorbu webových stránek a přiblížit znalosti v oblasti HTML značek, CSS nástrojů a frameworků, webového designu, měření rychlosti načítání webových stránek a jejich optimalizace.

Cílem praktické části této práce je zrealizovat analýzu výkonnosti vlastností a nástrojů CSS na odlišně složitých webových stránkách. Ty jsou vytvořeny za pomocí různých nástrojů a frameworků CSS, které jsou vhodné pro vznik optimalizovaného webu. Následně je provedeno zhodnocení výsledků a vyvození závěrečných doporučení, které nástroje, vlastnosti a frameworky CSS jsou rychlé, jaké mají největší vliv na rychlost načítání, které jsou přívětivé pro práci a které je tedy vhodné pro tvorbu webových stránek použít.

2.2 Metodika

Pro dosažení cílů této práce bude analyzována problematika práce s CSS nástroji a frameworky pro tvorbu webových stránek a pro analýzu jejich rychlosti načítání. Budou popsány potřebné teoretické znalosti v této oblasti. Dojde k vytvoření různých typů webových stránek za pomoci vhodných moderních nástrojů a frameworků CSS. Tyto webové stránky budou následně analyzovány z hlediska rychlosti jejich načítání pomocí měřícího nástroje. U vytvořených stránek se bude zkoumat vliv implementace nástrojů CSS, jak ovlivní rychlost načítání stránky daný nástroj CSS a jak bude čas ovlivněn při zpracování stránky přes framework. Bude se zkoumat, jaký vliv má použití nových vlastností a nástrojů oproti dříve používaným a také rychlost CSS při vyčištění kódu od přebytečných vlastností a hodnot. Měření budou provedena v různých prohlížečích a při kontrolované rychlosti internetového připojení. Všechna získaná a naměřená data budou hromadně zpracována a zhodnocena. Z těchto dat budou vyvozena doporučení, která mohou využít vývojáři webových stránek při své tvorbě a vytvořit tak optimalizované a rychlé webové stránky.

3 Teoretická východiska

3.1 World Wide Web

World wide web (WWW), běžně známý pod zkratkou web, označuje specifickou sekci internetu, která je přístupna uživatelům prostřednictvím webového prohlížeče. Je to spojení veškerých veřejně dostupných webových stránek, ke kterým uživatelé mohou přistupovat z různých zařízení. Přesto, že je mnohými význam World Wide Web chápán a zaměňován za výraz internet, jedná se ve skutečnosti o dvě rozdílné entity. World Wide Web tedy není synonymum internetu, jedná se o jeho část. internet je celosvětový systém vzájemně propojených počítačových sítí, který umožňuje uživatelům sdílet informace a komunikovat mezi sebou.[1]

Internet je globální sít s biliony počítačů a dalšího množství ostatních elektronických zařízení, propojených širokou škálou technologii, včetně optických vláken, satelitů nebo bezdrátových sítí. Jeho decentralizovaný model znamená, že není řízen žádnou centrální autoritou. Aby mezi sebou jednotliví uživatelé a zařízení mohli komunikovat, využívá internet sadu protokolů a standardů. Ty určují způsoby, jakými jsou data zasílána a přijímána. World Wide Web tedy tvoří velkou část internetu, nikoli však jedinou.[2]

World wide web lze přirovnat k extenzivní digitální knihovně, kdy jednotlivé webové stránky jsou uloženy na webových serverech po celém světe. Tyto webové stránky slouží jako základní kameny WWW. K vzájemnému propojení stránek používáme hypertextové odkazy. Ty nám umožňují navigaci z jednoho konkrétního bodu dokumentu nebo stránky do druhého. Aby mohl World Wide Web správně fungovat, je zapotřebí několik nezbytných komponent. Tyto komponenty spolu navzájem pracují, komunikují a doplňují se a umožnují tak uživateli interagovat s webovým obsahem.[3]

[1] ROUSE, Margaret. *Web*. Techopedia [online]. 2023 [cit. 2024-02-14]. Dostupné z: https://www.techopedia.com/definition/5613/web

[2] BERNERS LEE, Tim. *The World-Wide Web Initiative* [online]. In: . [cit. 2024-03-14]. Dostupné z: https://cds.cern.ch/record/2876913/files/930817.pdf

[3] AWATI, Rahul. *World Wide Web (WWW)*. TechTarget [online]. 2023. [cit. 2024-02-11] Dostupné z: https://www.techtarget.com/ whatis/definition/World-Wide-Web.

3.2 Základní komponenty WWW

Po obecném prozkoumání principu fungování webu a internetu, je nezbytné se zaměřit na jednotlivé stavební bloky, které tuto rozsáhlou digitální infrastrukturu vytvářejí, udržují a definují. Nyní budou tyto základní komponenty prozkoumány podrobněji a bude vysvětlena jejich role v tomto digitálním prostoru.

Celý proces začíná, když uživatel zadá do adresního řádku svého prohlížeče URL adresu webu. URL neboli „Uniform Resource Locator“ je jedinečný identifikátor pro konkrétní webovou stránku. Je taktéž označován jako webová adresa. URL se skládá z několika částí. Jedná se o typ protokolu, kterým jsou data posílána, název domény a konkrétní cestu. Tyto informace sdělují webovému prohlížeči jak a kde má zdroj načíst. Prohlížeč poté musí zjistit jaká IP adresa je spojenou s touto URL. K rozlišení různých počítačů, routerů a webových stránek na internetu je zapotřebí nějaký mechanismus. Řešení tohoto mechanismu nám nabízí právě IP adresa. Je to značka, která nám na internetu nebo v místní síti jednoznačně zařízení identifikuje. IP adresy tímto způsobem hrají zásadní roli ve fungování internetu. Následně je tedy potřeba tuto IP adresu přiřadit adrese URL. To může být provedeno buď pomocí serveru DNS nebo kontrolou mezipaměti cache, zda neobsahuje nějaké již existující záznamy. Jakmile se prohlížeči podaří nalézt danou IP adresu, naváže spojení s webovým serverem pomocí protokolu HTTP. V tomto procesu dále hrají důležitou roli následující subjekty a prvky.[4]

Transmission Control Protocol (TCP) představuje základní komunikační jazyk internetu. Jeho úloha spočívá v přijímání segmentů dat, jako jsou texty, obrázky nebo videa a jejich následné rozdělení na menší segmenty dat, takzvané pakety. Ty jsou s jeho pomocí přenášeny na místo určení, kde je další TCP vrstva schopna zajistit jejich příjem. Oproti HTTP protokolu, který se nachází v takzvané aplikační vrstvě, se TCP nachází v transportní vrstvě. Ve většině případů používá HTTP právě protokol TCP jako transportní protokol. TCP protokol nám definuje, jakým způsobem jsou data přenášena, formátována a jakým způsobem jsou zasílány pakety. Má tedy na starost samotný přenos dat. HTTP oproti tomu neřeší, jakým způsobem jsou data posílána, řeší pouze požadavky, které klient spustil a na které očekává od serveru odpověď. TCP nejsou samozřejmě jediné internetové protokoly sloužící k přenosu dat. Ukázkou dalšího může být například protokol UDP, který za určitých okolností nahrazuje TCP. Přenášené pakety jsou nepatrně menší, díky čemuž jsou vhodné pro různé aplikace,

[4] ROSS, David. *How the web works*. MDN Web Docs [online]. 2023 [cit. 2024-03-14]. Dostupné z: https://developer.mozilla.org/en- US/docs/Learn/Getting_started_with_the_web/How_the_Web_works?fbclid=IwAR1U-uiPtLXrrQTJotsNq4P_x%207aJCGZ9DZL_TT2bGvMnR-lLjlLB7i4J8Ss.

streamingové služby, hraní her a další internetovou komunikaci. Internet protokol (IP) má za úkol přesně definovat místo, kam mají být data přenesena a zajistit tak jejich konzistentní přenos a příjem. IP by bylo možné přirovnat k ekvivalentu systému GPS.[5]

Poskytovatel internetového připojení (ISP – internet service provider) funguje jako prostředník mezi klientem a serverem. Jakmile uživatel / klient vznese do prohlížeče požadavek na zobrazení určité webové stránky, prohlížeč neví, kde má tuto stránku hledat. V tomto momentu začíná práce poskytovatele internetu, který uživatele pomocí DNS spojí se správnou IP adresou webu.

Pro pochopení, jak IP a MAC adresa fungují, si můžeme představit, že internet vytváří jakousi paralelu s tradičními poštovními službami. Místo klasické adresy tvořené z ulice, čísla popisného a města máme IP adresu. Naopak MAC adresa nám symbolizuje jméno. Oba tyto prvky spolu navzájem spolupracují a zajišťují tak efektivní přenos dat mezi stranami. Celá struktura internetu je tvořena odlišnými sítěmi. Každou tuto síť si můžeme představit jako skupinu objektů, které jsou zastřešovány již dříve zmíněným ISP, neboli poskytovatelem internetového připojení. Jakmile od poskytovatele získáme jeho služby, staneme se součástí této specifické sítě a následně nám to umožní přístup k dalším sítím. Každý poskytovatel internetového připojení pod sebou spravuje určitou množinu IP adres. Jakmile se přihlásíme k odběru jeho služeb, je nám přidělena konkrétní IP adresa. Pomocí této unikátní IP adresy nás ostatní dokážou v síti identifikovat a nasměrovat nám potřebná data. MAC adresa představuje unikátní identifikátor síťového zařízení. Tato adresa je přidělována na síťovou kartu přímo výrobcem, z tohoto důvodu je taktéž označována jako fyzická adresa.

DNS je zkratka označující systém doménových jmen. Tento systém funguje jako digitální adresář pro webové stránky. Když do adresního řádku prohlížeče napíšeme název domény webu, na který se chceme dostat, prohlížeč se nejdříve spojí s DNS serverem, aby zjistil IP adresu spojenou s daným webem (názvem webu). Tento krok je nezbytný, protože prohlížeč musí zajistit přesnou adresu serveru, na kterém je stránka uložena.[6]

---

[5] HIWARALE, Uday. *A brief overview of the TCP/IP model, SSL/TLS/HTTPS protocols and SSL certificates*. Medium [online]. 2020 [cit. 2024-03-14]. Dostupné z: https://medium.com/jspoint/a-brief-overview-of-the-tcp-ip-model-ssl-tls-https-protocols-and-ssl-certificates-d5a6269fe29e

[6] KASIREDDY, Preethi. *How the Web Works*. Freecodecamp [online]. 2015 [cit. 2024-03-14]. Dostupné z: https://www.freecodecamp.org/news/how-the-web-works-a-primer-for-newcomers-to-web-development-or-anyone-really-b4584e63585c/?fbclid=IwAR3_lgf1iqrbNR3eP_QmkO%20SMU2X-whcccbvF042oLdTw-2tPFtCz8KkGBHk.

3.3 Základy a technologie webového rozhraní

Webové stránky se dají rozdělit do dvou základních komponent. Obě tyto komponenty jsou nedílnou součástí každého webu a zajišťují správné fungování webové stránky. První, takzvaná front-end část, poskytuje uživatelské rozhraní. Tedy představuje veškeré viditelné prvky na stránce, jako je písmo, obrázky nebo layout, díky kterým se může uživatel na stránkách dobře orientovat. Druhá komponenta, takzvaný back-end, zahrnuje skrytý rámec nezbytný pro funkčnost front-endu. Stará se o to, aby všechny požadavky putující od uživatele mohly být přeneseny na server skrze webovou interakci.

Front-end vývojáři jsou zodpovědní za vytváření vizuálních prvků webových stránek se zaměřením na aspekty interakce s uživatelem jako jsou barvy, rozvržení a písma. K tvorbě webových stránek se používají tři základní webové technologie. Dalo by se je označit za tři základní stavební kameny webového front-endu. Těmi jsou HTML, CSS a JavaScript. HTML vytváří obsah a strukturu, CSS přidává této struktuře design a JavaScript vytváří interaktivní funkce.

Oproti tomu back-end vývojáři se soustředí na vytvoření neprůstřelné infrastruktury, která zajišťuje bezproblémové fungování webových stránek. Soustředí se na databáze, server, API a architekturu celého webu, tedy komponenty, které obvykle zůstávají skryté před zraky uživatelů počítačů, fungují pod povrchem webové stránky. S jejich pomocí je možný správný tok, uchovávání a pochopení dat. V tomto případě je nezbytná znalost programovacích jazyků jako je Java, Python, Ruby nebo PHP. Vývojáři také využívají nástroje jako je SQL dotazovací jazyk, a to ke správě a uchovávání dat. Tato práce se zaměřuje na vliv vlastností a nástrojů CSS na rychlost načítání webových stránek, tedy na výše zmíněnou první neboli front-endovou část webu. V následujícím textu této práce budou stručně představeny nejpodstatnější koncepty jazyka CSS.[7]

3.4 Model klient-server

Pro pochopení samotné podstaty fungování a ostatních vlastností jako je rychlost načítání a proces zobrazení webové stránky, je nutné definovat model klient – server. Základní princip fungování webové stránky spočívá v komunikaci mezi webovým prohlížečem na straně

[7] SIMMONS, Liz. *Front-End vs. Back-End*. Computerscience [online]. 2023 [cit. 2024-03-14]. Dostupné z: https://www.computerscience.org/bootcamps/resources/frontend-vs-backend/

uživatele (klientem) a webovým serverem, na kterém je stránka uložena. Tento koncept je v oblasti vývoje webu označován jako model klient-server. Model představuje základní stavební kámen fungování webu a jeho pochopení otevírá možnosti pro správnou, efektivní a optimalizovanou tvorbu webových stránek.[8]

Klient je označení pro webový prohlížeč, tedy aplikaci nainstalovanou na uživatelském zařízení. Příkladem mohou být Microsoft Edge, Safari, Tor nebo Google Chrome. Jejich primární funkcí je interpretovat uživatelskou interakce do požadavků odeslaných na webový server. Ten poskytuje samotnou webovou stránku, aniž bychom ji museli mít lokálně uloženou na našem zařízení. I přesto, že k přístupu na web je používán webový prohlížeč, klientem může být označováno celé zařízení, tedy mobilní telefon, tablet nebo počítač, se kterým je vše ovládáno. Každé takto připojené zařízení má přidělenou jedinečnou identifikaci – IP adresu, pomocí které mohou být zařízení jednoznačně identifikovatelná.

Webový server je druhý klíčový prvek, jehož primární funkcí je reagovat a zpracovávat požadavky putujících od klientů. Podstata webového serveru spočívá v jeho schopnosti ukládat, zpracovávat a doručovat obsah webových stránek uživatelům (klientům). Po odeslání webového požadavku ze strany klienta, spravuje požadavek jedna ze serverových aplikací. Ta ověřuje platnost požadavku a určuje dostupnost požadovaných informací na serveru. Po potvrzení aplikace webového serveru pokračuje ve zpracování a následném doručení požadované informace webovému klientovi. Webový klient a server spolu komunikují pomocí takzvaných protokolů. Mezi protokoly řadíme například HTTPS protokol. Komunikace s jinými servery však může být prováděna i jinými protokoly.[9]

3.5 Webový prohlížeč

Nejpodstatnější funkcí webového prohlížeče je zobrazení uživatelem vybraného webového zdroje (webové stránky). Tím je ve většině případů HTML dokument, avšak takto může být zobrazen například i PDF dokument nebo třeba fotografie. Způsob, jakým prohlížeč interpretuje tyto dokumenty je popsán v HTML a CSS specifikacích, které jsou dohlíženy W3C

[8] TANENBAUM, Andrew a David WETHERALL. *Computer Networks* [online]. 5th edition. Pearson, 2010 [cit. 2024-01-14]. ISBN 978-0132126953. Dostupné z: https://learning.oreilly.com/library/view/computer-networks-fifth/9780133485936/

[9] KASIREDDY, Preethi. *How the Web Works*. Freecodecamp [online]. 2015 [cit. 2024-03-14]. Dostupné z: https://www.freecodecamp.org/news/how-the-web-works-a-primer-for-newcomers-to-web-development-or-anyone-really-b4584e63585c/?fbclid=IwAR3_lgf1iqrbNR3eP_QmkO%20SMU2X-whcccbvF042oLdTw-2tPFtCz8KkGBHk.

(World Wide Web Consortium) organizací pro standardizaci webu. Toto v minulosti nebylo pravidlem, a tak si prohlížeče často vyvíjely vlastní rozšíření, což vedlo ke značným problémům s kompatibilitou napříč různými prohlížeči. V současné době je v desktopové verzi používáno ve většině případů pět webových prohlížečů. Těmi jsou Chrome, Safari, Microsoft Edge, Opera a Firefox. V mobilní verzi to jsou prohlížeče Chrome, Safari, Samsung Internet, Opera a UC Browser. Oproti standardizaci webu, standardizace webového prohlížeče není pevně stanovena, existují však určitá doporučení. Mezi standardní a běžné prvky, které jsou a měly by být součástí většiny prohlížečů, patří adresní řádek pro vložení URL, tlačítka pohybování se zpět a vpřed společně s tlačítkem pro obnovení stránky, oblíbené stránky a záložky, domovské tlačítko. Je tu pak i mnoho dalších prvků, které jsou specifické pro daný prohlížeč.[10]

Webový prohlížeč lze rozdělit na dvě základní části a tím je část front-end a back-end. Front-end prohlížeče, který přímo komunikuje s uživatelem, obsahuje prvky grafického uživatelského rozhraní (GUI). Těmi jsou například adresní řádek nebo navigační tlačítka. Kromě toho je front-end zodpovědný za vykreslování webových stránek, prezentaci obsahu, obrázků a interaktivních prvků na obrazovce uživatelova zařízení. Mezi základní jazyky pro front-endovou část webu patří HTML, CSS a JavaScript.

Komplikované zákulisí webové stránky, které není na první pohled uživateli dostupné se nazývá back-end. V této části se řídí celá komunikace mezi webovým prohlížečem a serverem. Back-end zajištuje správnou interpretaci HTML, CSS a JavaScriptu, aby se zajistila správná funkčnost jednotlivých prvků. Kromě toho má back-end za úkol síťová připojení, řeší různé protokoly jako je HTTP a HTTPS nebo se stará o zabezpečení, šifrování a ověřování certifikátů. Kromě těchto dvou hlavních částí lze fyzická struktura webového prohlížeče shrnout do sedmi základních komponentů:

1. **Uživatelské rozhraní** – Uživatelské rozhraní se dá jednoduše představit jako pracovní plocha, kde se odehrává veškerá interakce. Zahrnuje prvky, jako jsou záložky, adresní řádek, do kterého se zadávají názvy webových stránek, tlačítka pro přechod tam a zpět, karty pro provádění mnoha úkonů najednou a nabídky pro vyhledání různých věcí v prohlížeči.
2. **Jádro prohlížeče** – Jádro prohlížeče je mozek prohlížeče, který má na starost veškeré důležité funkce. Koordinuje správný tok informací mezi uživatelským rozhraním,

[10] BODNAR, Danielle. *What Is a Web Browser?* Avast [online]. 2023 [cit. 2024-01-05]. Dostupné z: https://www.ava st.com/c-what-is-a-web-browser

jádrem prohlížeče a ostatními komponenty. Zajišťuje, aby vše, co uživatel na stránce udělá, proběhlo hladce.

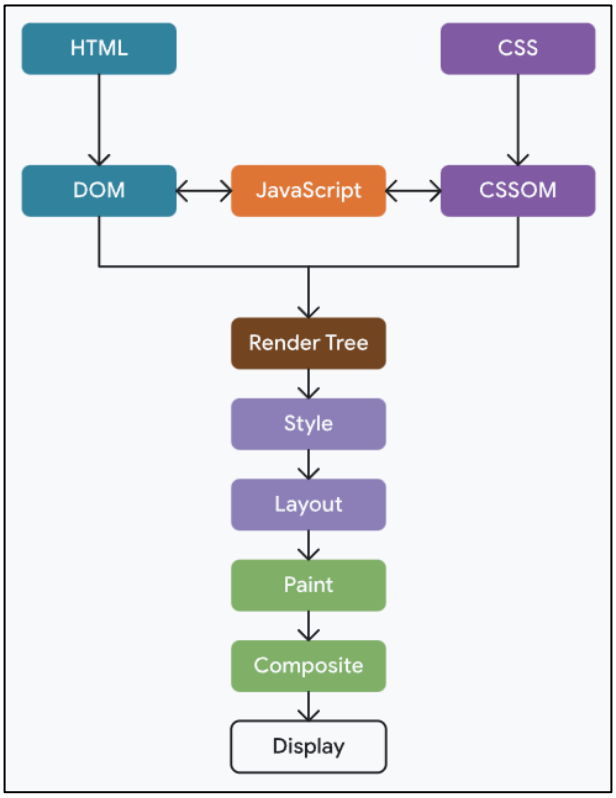
3. **Vykreslovací jádro** – Vykreslovací jádro vykresluje / vyobrazuje webové stránky na obrazovku. Převádí kód stránky – HTML, CSS a JavaScript do vizuální formy. Dokáže interpretovat HTML kód, aplikuje CSS styly pro vykreslení layoutu a vzhledu stránky a spustí případný JavaScript pro přidání více dynamických prvků. Vykreslovací jádro obsahuje veškeré potřebné prvky a logiku pro vykreslení webové stránky od HTML kódu až po první zobrazený pixel na obrazovce.
4. **JavaScript interpret** – Interpret JavaScriptu umožní to, že kód JavaScriptu bude správně spuštěn a zajistí, aby stránka byla schopna správně reagovat na uživatelské požadavky.
5. **UI back-end** – Tato část se používá k vytvoření základních prvků, jako jsou combo boxy a okna. Zobrazuje univerzální rozhraní, které není vázáno na žádnou konkrétní webovou stránku.
6. **Datové uložiště** – Pro možnost lokálního ukládání rozmanitých dat, například souborů cookies, je nutné, aby webový prohlížeč podporoval různé mechanismy ukládání dat, včetně WebSQL, IndexedDB, FileSystem a dalších. Tato datová vrstva je trvalého charakteru.
7. **Síťový modul** – Síťový modul v prohlížeči spravuje veškeré aspekty síťové komunikace. Provádí úkony, jako je propojení URL adres webových stránek na adresy IP, odesílání HTTP požadavků na webové servery a zpět, vytváření síťových odkazů a řízení příjmu a zpracování dat. Tento modul je klíčový k získávání veškerých zdrojů (HTML, CSS, obrázků a různých souborů) ze serverů. Tyto prostředky následně přenese do renderovacího (vykreslovacího) jádra.[11]

3.6 Proces vykreslování webové stránky

Před samotným vyobrazením prvního pixelu na uživatelově obrazovce musí prohlížeč projít několika nezbytnými kroky. Sekvence těchto postupů a kroků se nazývá kritická vykreslovací cesta, která hraje klíčovou roli při vykreslování webové stránky. Kritická

[11] GARSIEL, Tali. *How browsers work*. Web dev [online]. 2011 [cit. 2023-12-10]. Dostupné z: https://web.dev/articles/howbrowserswork?fbclid=IwAR0s9408bs99IStDtpZWnjVSiRCvXW63lIBCRmi-cjOM7EpVum2YtWz7lWw#the_browsers_high_level_structure

vykreslovací cesta představuje sled kroků, kterými se prohlížeč řídí při přeměně HTML, CSS a JavaScriptu na první viditelné pixely na obrazovce. Optimalizace této kritické cesty dokáže signifikantně zvýšit výkon vykreslování. Kritická cesta se skládá z několika klíčových prvků. Těmito prvky jsou Document Object Model (DOM), CSS Object Model (CSSOM), vykreslovací neboli renderovací strom a layout.



Obr. č. 1: Kritická vykreslovací cesta[12]

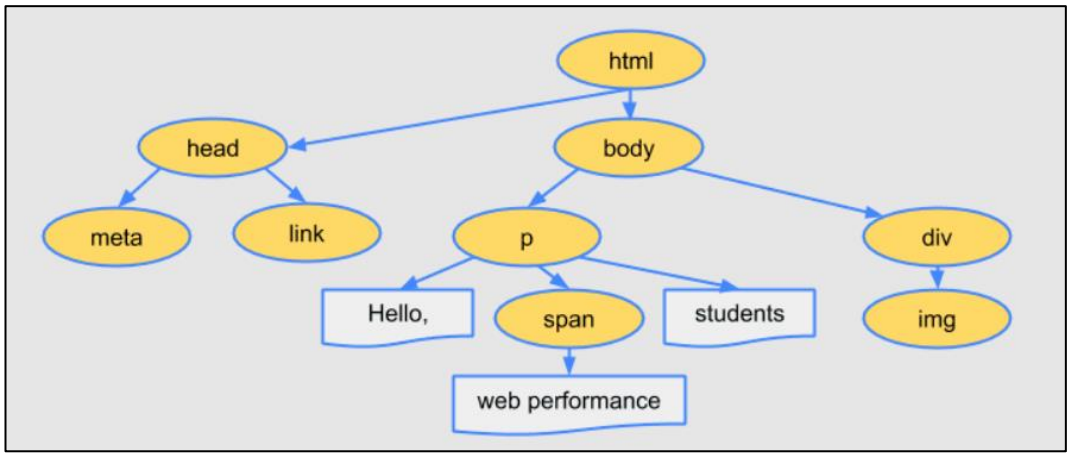
Když uživatel zadá do adresního řádku adresu webu, prohlížeč si od webového serveru, kde je stránka uložena, vyžádá HTML dokument. Ten prohlížeči doputuje v binárním formátu, což se dá považovat za obyčejný textový dokument, který má ve své hlavičce specifikováno, že se jedná o HTML kód. Díky znalosti těchto informací, může prohlížeč tento dokument v binárním formátu převést do čitelného souboru podobného textovému dokumentu.[13]

Document Object Model neboli DOM je datovou reprezentací objektů, které spolu vytváří strukturu a obsah webové stránky. Je to objektivně orientovaná reprezentace webové

[12] LENUKA, Melia. *Understanding the critical path.* WebDev [online]. 2023 [cit. 2024-02-05]. Dostupné z: https://web.dev/learn/performance/understanding-the-critical-path?fbclid= IwAR0rAS6h7N4aWtKDb9DMAvC6cq8wuKvgkvU_I-BqSfEyUTckgFaKfB1Q_-E

[13] SPENCER, Corvin. *Critical rendering path*. Mdn web docs [online]. 2023[cit. 2024-01-14]. Dostupné z: https://developer.m ozilla.org/en-US/docs/Web/Performance/Critical_rendering_path.

stránky. Při čtení HTML kódu, který prohlížeč obdržel od webového serveru, vytváří prohlížeč z každého HTML elementu JavaScriptový objekt (většinou je s ním manipulováno právě JavaScriptem, ale není to nezbytně nutné) nazývaný Node. Tímto způsobem jsou postupně všechny HTML elementy/prvky transformovány do jednotlivých objektů. Vzhledem k tomu, že každý prvek v HTML má jedinečné vlastnosti, vzniká Node objekt z různých tříd (konstruktorů). Jako příklad lze uvést, že objekt Node pro prvek div je vytvořen pomocí HTMLDivElement, který zdědil třídu Node. Poté, co byly tímto způsobem vytvořeny všechny DOM Node objekty, prohlížeč z nich vytvoří podobnou stromovou strukturu jako z HTML dokumentu nazývanou DOM Tree. Ta prohlížeči pomůže při efektivním zvládnutí renderování stránky. Je třeba zdůraznit, že DOM není součást samotného JavaScriptu. Je to webové API pro tvorbu stránek. Zajišťuje efektivní renderování webové stránky a také snadnou manipulaci se stránkou, které přidává dynamiku. Pokud je nějaký element nutné pozměnit, přidat nebo smazat je tak provedeno právě díky DOMu.[14]



Obr. č. 2: DOM[15]

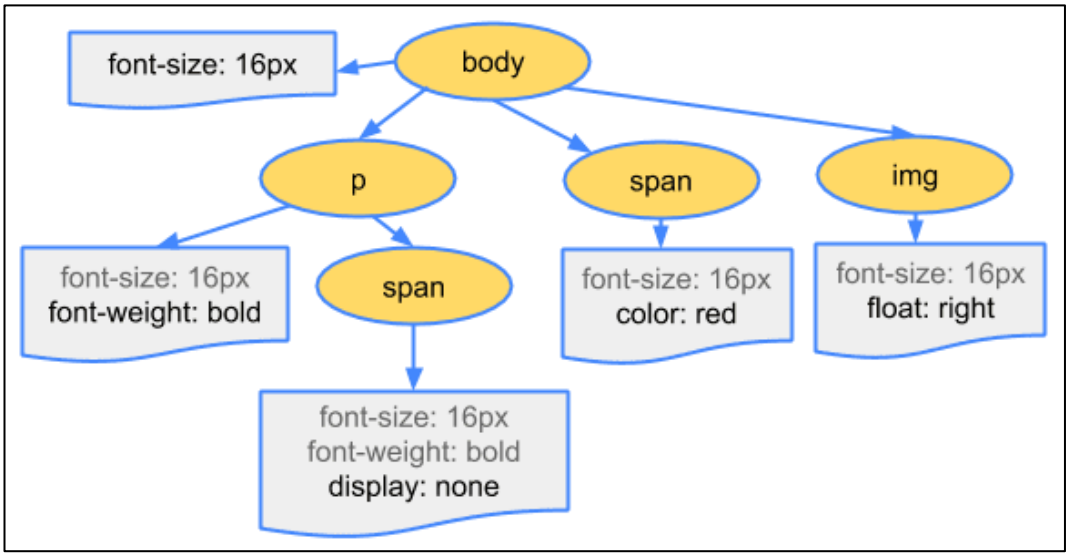
Zatímco je prohlížečem vytvářen DOM, prohlížeč si načte CSS soubor. Ten může být připojen na HTML soubor několika způsoby. Stejně jako u HTML tak i v případě CSS informace ze serveru přichází v bajtech a na straně klienta bude poté převeden na samotný čitelný soubor se styly. Způsob, jakými jsou CSS styly aplikovány na jednotlivé HTML

[14] HIWARALE, Uday. *How the browser renders a web page? — DOM, CSSOM, and Rendering*. Medium [online]. 2019 [cit. 2024-01-21]. Dostupné z: https://medium.com/jspoint/how-the-browser-renders-a-web-page-dom-cssom-and-rendering-df10531c9969

[15] GRIGORIK, Ilya. *Constructing the Object Model.* Web Dev [online]. 2014 [cit. 2024-01-05]. Dostupné z: https://web.dev/articles/critical-rendering-path/constructing-the-object-model?fbclid=IwAR30SfjepRbMUaQ0FI2Ccc0xE8O_pgXmxOB1KAdPesLpn5uUbym-499Bpls

elementy, respektive na DOM objekty, označujeme jako CSS selektory. Selektory mohou být použity jak JavaScriptem, tak CSS. Umožnují zacílit na jednotlivé HTML elementy a změnit jejich vzhled, a to na základě jejich vlastností, atributů, stavů, typů anebo umístění.

Po správném načtení CSS souboru s ním prohlížeč může začít korektně pracovat a vytvoří z něj takzvaný CSSOM. CSSOM je označení pro CSS Object Model, což je stejně jako DOM dokument se stromovou strukturou uspořádání. Každý Node z této stromové struktury obsahuje CSS vlastnosti, které mu jsou přímo přiřazeny CSS selektory. Je třeba si uvědomit, že CSSOM neobsahuje elementy, které nejsou vyobrazeny na obrazovce, tedy například <meta> nebo <title>. Další podstatný fakt je ten, že každý prohlížeč má své defaultní CSS styly, jakými zobrazuje jednotlivé elementy. Než prohlížeč zkonstruuje node, tyto defaultní vlastnosti přepíše. Pokud určitá CSS vlastnost není pro konkrétní html prvek explicitně definována vývojářem nebo prohlížečem, automaticky převezme výchozí hodnotu specifikovanou ve W3C standardu. V opačném případě řada vlastností, které se týkají práce s odstavci nebo textem takzvaně dědí své vlastnosti od svých rodičovských elementů. [16]



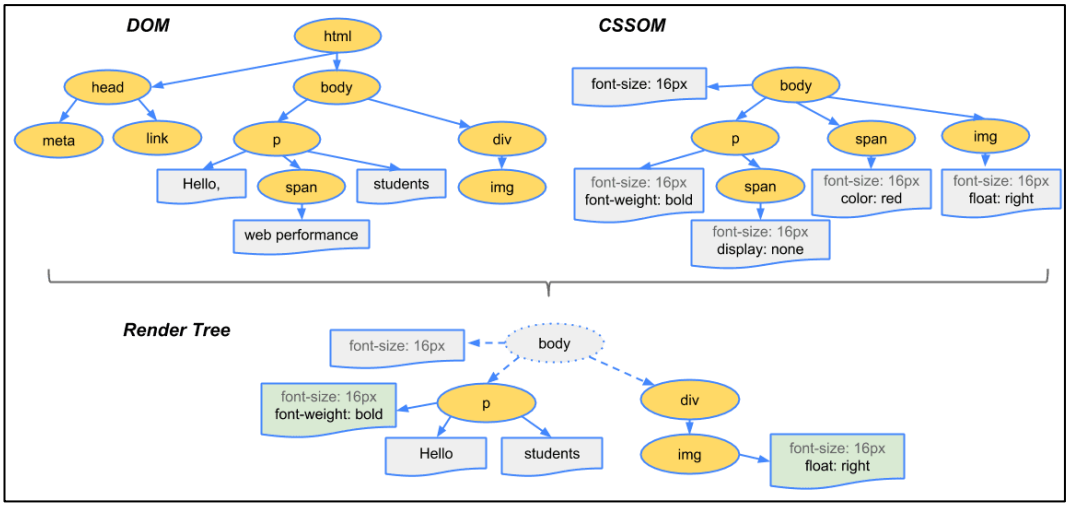
Obr. č. 3: CSSOM[17]

Před konečným vykreslením jednotlivých prvků na obrazovku musí prohlížeč vypočítat layout jednotlivých viditelných elementů a následně je vykreslit. Toho je dosaženo za pomoci

[16] HIWARALE, Uday. *How the browser renders a web page?* — DOM, CSSOM, and Rendering. Medium [online]. 2019 [cit. 2024-01-21]. Dostupné z: https://medium.com/jspoint/how-the-browser-renders-a-web-page-dom-cssom-and-rendering-df10531c9969

[17] GRIGORIK, Ilya. *Constructing the Object Model.* WebDev [online]. 2014 [cit. 2024-01-05]. Dostupné z: https://web.dev/articles/critical-rendering-path/constructing-the-object-model?fbclid=IwAR30SfjepRbMUaQ0FI2Ccc0xE8O_pgXmxOB1KAdPesLpn5uUbym-499Bpls

renderovacího stromu. Ten je vytvořen spojením DOM a CSSOM a opět nese stromovou strukturu. Dokud není renderovací strom vytvořen, nemůže být na obrazovku nic vykresleno. Renderovací strom je reprezentací toho, co následně na obrazovce uvidíme. Renderovací strom neobsahuje elementy, které nezabírají žádné pixely. Například prvky, které mají nastavené rozměry 0px, nebo vlastnost display: none, nebudou v renderovacím stromu obsaženy. Na rozdíl od DOM API, který nám umožní přistupovat ke stromu DOM, je CSSOM uživateli skryto. Právě proto, že prohlížeč vytvoří renderovací strom kombinací těchto dvou, získá uživatel možnost CSS vlastnosti měnit.[18]



Obr. č. 4: Renderovací strom[19]

DOM a CSSOM je k dispozici, renderovací strom je vytvořen. Nyní je třeba vše přenést na obrazovku. Toto nazýváme renderovací proces, který se skládá ze třech kroků. První z nich je takzvaný layout. Layout má za úkol veškerý proces kalkulace a umístění jednotlivých prvků na stránce. Vypočítá nám dle velikosti jednotlivých elementů jejich rozložení na stránce, vzdálenosti a pozici. V tomto kroku nemáme na obrazovce stále zobrazen ani jeden pixel. Druhý krok renderovacího procesu se nazývá painting, což je samotné „vymalování“ jednotlivých prvků. To znamená, že jednotlivé prvky už budou na obrazovce barevně viditelné dle svých přidělených CSS vlastností. Poslední fází renderovacího procesu je compositing, což je technika, která zahrnuje rozdělení různých prvků webové stránky do odlišných vrstev,

[18] GARSIEL, Tali. *How browsers work*. Web.dev [online]. 2011. [cit. 2024-01-14]. Dostupné z: https://web.dev/articles/howbrowse rswork#render_tree_construction.
[19] GRIGORIK, Ilya. Render-tree Construction, Layout, and Paint. Web dev [online]. [cit. 2024-03-14]. Dostupné z: https://web.dev/articles/critical-rendering-path/render-tree-construction?fbclid=IwAR2dYMZ4TckMEia0SMCnTTeFM-i0jhJxNR6agimLw4EDXdqZHZf2I5Hhktc..

rastrování každé vrstvy nezávisle a jejich následné sloučení do podoby kompletní stránky. Všechno výše zmíněné, od tvorby DOMu až po zobrazení prvního pixelu na obrazovce, má za úkol vykreslovací jádro prohlížeče, které v sobě obsahuje vše, co je potřebné pro generování DOMu z HTML dokumentu, který přijde z webu i vše pro vyobrazení celé webové stránky. Webové prohlížeče Google Chrome, nebo Opera používají jako jádro Blink. Safari používá jako své jádro WebKit a Mozilla Firefox je postaven na jádře zvaném Gecko.[20]

3.7 CSS

Kaskádové styly (z anglického Cascading Style Sheets – CSS) jsou stylovací jazyky, které nám umožnují definovat, jakým způsobem mají být jednotlivé prvky jazyka HTML nebo XML vykresleny na obrazovku. Přesněji řečeno nám umožnují nadefinovat velikost, barvu, text, okraje, umístění a mnoho dalších vlastností ke všem HTML elementům na stránce. CSS patří mezi základní jazyky pro tvorbu webu a je standardizován W3C specifikací. Ke správnému pochopení, jak CSS funguje a jak jsou jednotlivé styly elementům html přiřazeny, je zapotřebí si vysvětlit terminologii, která se při aplikaci CSS používá. [21]

Základní prvek, se kterým se při aplikaci CSS stylů pracuje, je selektor. Pomocí selektorů přiřazujeme jednotlivé námi definované vlastnosti. Selektor si můžeme jednoduše představit větou „Co chceme upravit“. Mezi nejpoužívanější selektory patří selektory založené na HTML elementech, kdy se přímo uvede název elementu, kterému chceme přiřadit styl. Dalšími selektory může být například selektor třídy nebo id selektor. Jakmile máme selektor, tedy zvolili jsme si co chceme upravit, pokračujeme dále tím, jak daný HTML element, třídy nebo id chceme upravit. Jinak řečeno už si přesně zvolíme, jak chceme, aby daný prvek vypadal. Toto se obecně označuje jako deklarace. Deklarace lze rozdělit na dvě části a těmi jsou vlastnost a hodnota. Každá hodnota je vlastnosti přiřazena dvojtečkou. Jednotlivé vlastnosti můžou být například color nebo font-size. Hodnotou je už pak konkrétní „vizuální stav“, kterého může vlastnost nabývat. Hodnota vlastnosti color může být tedy například red a hodnota vlastnosti font-size může být například 12px. Jakmile je tedy selektorem zvolen html prvek, který má být

[20] HIWARALE, Uday. *How the browser renders a web page? —* DOM, CSSOM, and Rendering. Medium [online]. 2019 [cit. 2024-01-21]. Dostupné z: https://medium.com/jspoint/how-the-browser-renders-a-web-page-dom-cssom-and-rendering-df10531c9969

[21] MEYER, Eric A. *CSS: the definitive guide: web layout and presentation* [online]. Fifth edition. Beijing: O'Reilly, 2023 [cit. 2024-01-11]. ISBN 978-1098117610. Dostupné z: https://learning.oreilly.com/library/view/css-the-definitive/9781098117603/

upraven, je napsána deklarace složená z vlastností, která představuje, jaký stav nebo vlastnost má být u prvku upravena a hodnota podle toho, co vlastnosti bude přiřazeno. Deklarace společně se selektorem je nazývána jako pravidlo. Počet CSS vlastností, které mohou být jednotlivým elementům, třídám, případně ostatním prvkům přiřazeny je přibližně několik stovek (cca 500). Ne všechny vlastnosti jsou ovšem prohlížeči podporovány. Pokud je řeč o hodnotách, těch je prakticky neomezeně.

Pseudotřída představuje selektor, který identifikuje prvky v určitém stavu. Napomáhá nám tedy identifikovat například první či poslední prvek obsažený v rodičovském prvku, nebo prvek, nad kterým se zrovna nachází ukazatel myši. Pseudotřída vždy začíná dvojtečkou. Fungování všech pseudotříd je v podstatě na podobném principu. Vždy se snaží zacílit na určitou část dokumentu, který je v nějakém stavu. Příkladem pseudotříd může být například :last-child, :only-child nebo :invalid. Některé pseudotřídy se stávají relevantními až tehdy, pokud uživatel interaguje s nějakým atributem. Tyto pseudotřídy, označované jako dynamické pseudotřídy, napodobují chování, jako by byla k prvku přidána třída během interakce uživatele. Mezi příklady patří :hover nebo také :focus. Pseudoelementy mají obdobnou funkci. Jejich princip oproti pseudotřídám spočívá v tom, jako by se do struktury webové stránky přidal zcela nový HTML prvek, spíše než aplikace třídy na existující prvky. Pseudoelementy začínají dvojitou dvojtečkou ::. Jako příklad pseudoelementu poslouží ::before. Pokud by měl být například vybrán první řádek nějakého odstavce, mohl by být obalen do atributu nebo . Tento způsob by však fungoval pouze tehdy, pokud by byl rodičovský element stejně široký jako odstavec, v jiných případech by to způsobovalo kolizi. Pro tento případ je právě vhodné použití pseudoelementu ::first-line, který problém spolehlivě vyřeší. [22]

Jedna z nejpodstatnějších věcí, pro tvorbu responzivního webu jsou media queris. Media queries dávají možnost zacílit a přizpůsobit aplikované styly například podle toho, z jakého zařízení uživatel přichází, jaký webový prohlížeč používá, jak široký je jeho display, nebo zda je jeho zařízení dotykové nebo bezdotykové. Toto je klíčová vlastnost CSS, za pomoci které můžeme přizpůsobit vzhled a rozložení webové stránky pro jednotlivá zařízení takovým způsobem, abychom maximalizovali uživatelův prožitek (UX). Media query je tvořena dvěma

[22] MEYER, Eric A. CSS: the definitive guide: web layout and presentation [online]. Fifth edition. Beijing: O'Reilly, 2023 [cit. 2023-11-08]. ISBN 978-1098117610. Dostupné z: https://learning.oreilly.com/library/view/css-the-definitive/9781098117603/

složkami. Těmi jsou typ media a vlastnost média. Ty mohou být kombinovány pomocí různých logických operátorů.[23]

3.8 Modulární CSS

CSS je poměrně intuitivní jazyk a naučit se ho ovládat by nemělo obnášet velké komplikace. Toto však platí pouze tehdy, jde-li o menší projekt nebo projekt v začátcích. Jakmile se projekt začne svou velikostí a komplexností rozrůstat, práce se stává čím dál tím více komplikovaná. A právě v tomto kroku přijde metodologie psaní CSS vhod. Modulární CSS neboli také metodika pro organizaci CSS kódu, představuje strukturování kaskádových stylů (CSS) modulárním, nezávislým a opakovaně použitelným způsobem. Tento přístup má za cíl zlepšit udržovatelnost kódu, škálovatelnost a jeho znovupoužitelnost. Důležitý aspekt klade na zlepšení spolupráce mezi vývojáři a na týmovou práci na projektu. Rozdělením stylů do menších samostatných komponent neboli modulů se případné úpravy stanou lépe editovatelnými a zredukuje se tak riziko nezamyšlených chyb. Modulární CSS nám napomáhá k systematické organizaci stylů a podporuje tak efektivnější vývojový proces. Vývojáři mohou vytvářet nezávislé moduly pro konkrétní komponenty, což usnadňuje opětovné použití kódu v různých částech webu, zefektivňuje pracovní postup vývoje, ale také zvyšuje jasnost a srozumitelnost kódu. V praxi je možné se setkat s několika takovými metodologickými přístupy.[24]

Pokud si zvolíme OOCSS jako jednu z možných CSS metodologií, je třeba dodržet dvě základní pravidla. Je to pravidlo oddělení struktury a kůže (anglicky structure and skin). Pod strukturou si můžeme představit vlastnosti zahrnující velikost a pozicování jednotlivých elementů jako jsou size, margin nebo paddings, jednoduše by se dalo říci, že se jedná o základní vlastnosti, které mají jednotlivé elementy společné a můžeme tak tento styl přiřadit všem elementům. Pod pojmem kůže si může představit vizuální aspekty elementů jako jsou color a shadows. Tyto vlastnosti už má každý element odlišné. Druhé pravidlo, které se při OOCSS

[23] CASTRO, Elizabeth a HYSLOP, Bruce. In: *HTML5 a CSS3: názorný průvodce tvorbou WWW stránek*. Brno: Computer Press, 2012. ISBN 978-80-251-3733-8.
[24] VANDEHEY, Scott. What is Modular CSS? Space Ninja [online]. 2018 [cit. 2023-11-09]. Dostupné z: https://spaceninja.com/blog/2018/what-is-modular-css/

používá, je oddělení kontejneru a obsahu. To v realitě znamená, že je nutné se vyhnout používání pseudotříd a také omezit používání vnořených stylů.[25]

Další metodologický koncept je BEM, používaný k tvorbě přehledného CSS. BEM je zkratka tří slov. Block, Element a Modifier. Celý koncept BEM je založen právě na těchto třech slovech představujících jednotlivé úrovně pojmenovávání. Podle těchto třech úrovní je striktně dodržována jmenná konvence a nemůže tak dojít ke kolizi. Blok představuje hlavní komponentu na stránce. Blokem může být navigace, článek, side bar nebo jakýkoli jiný hlavní prvek. Element je pak každá jednotlivá část bloku. Tedy side bar, který představuje blok může obsahovat například nějaký paragraf, nadpis a tlačítka. Všechny tyto prvky představují elementy. Jako poslední je zde Modifier. Toto je specifický prvek, který je možné použít současně se zadefinováním bloku nebo elementu a přestavuje tak specifickou třídu jeho změny.[26]

Atomic CSS představuje metodologii, která klade důraz na využívání malých, jednoúčelových tříd. Každá třída v ACSS aplikuje konkrétní styl na ten prvek na webové stránce, kterému je přiřazena. Tento přístup značně usnadňuje tvorbu a správu stylů, neboť každá třída slouží právě jednomu konkrétnímu účelu. Jádrem celého ACSS je tedy používání separátních individuálních tříd, které aplikují specifický styl na daný prvek. Třídy jsou pojmenovány podle stylu, který aplikují, například třída .bg-blue je nazvána podle background-color: #007bff nebo třída .cl-white symbolizuje vlastnost color: white. Je důležité zmínit, že se v praxi používá kombinace těchto přístupů. Tedy použití jednotlivých metodik není výlučné, ale naopak se vzájemně doplňují a usnadňují tak webovou tvorbu.[27]

3.9 Drahé vlastnosti CSS

Určité CSS vlastnosti mohou ovlivnit načítání webových stránek a jejich celkový výkon více než ostatní. Jejich nesprávné použití tak může webovou stránku negativně ovlivnit, zpomalit ji a zhoršit její interaktivitu s uživatelem. Tyto vlastnosti jazyka CSS se dají označit

[25] MICHÁLEK, Martin. *OOCSS: objektové psaní CSS*. Vzhurudolu [online]. 2017 [cit. 2024-01-14] Dostupné z: https://www.v zhurudolu.cz/prirucka/oocss..
[26] MICHÁLEK, Martin. *BEM: Pojmenovávací konvence pro třídy v CSS*. Vzhurudolu [online]. 2017 [cit. 2024-01-14]. Dostupné z: https://www.vzhurudolu.cz/prirucka/bem.
[27] POLACEK, John. *Let's Define Exactly What Atomic CSS is*. Css-tricks [online]. 2017 [cit. 2024-01-14]. Dostupné z: https://css-tricks.com/lets-define-exactly-atomic-css/.

souhrnným názvem jako drahé vlastnosti CSS. V následujícím textu jsou popsány některé z těchto drahých, respektive nejnáročnějších CSS vlastností a způsoby, jak je optimalizovat.

Box-Shadow – Použití vlastnosti box-shadow je velmi oblíbený způsob, jak přidat stínový efekt k jednotlivým elementům. Jeho vliv na výkon může být ovšem poměrně značný. Zejména pak, je-li aplikován na velké množství elementů nebo s velkým rádiusem rozostření (hodnota blur-radius). Poté může dojít k poměrně znatelnému zpomalení načítání webové stránky. Řešení optimalizace této vlastnosti se nabízí hned několika způsoby. V první řadě je třeba se zaměřit na eliminaci množství elementů s touto vlastností. Pokud se tuto vlastnost rozhodneme použít, měla by být použita pouze decentně a s malým rádiusem rozostření. Dále je vhodné použít pouze jednotnou barvu, které také usnadní vykreslování. V případě používání této vlastnosti na vnitřní stíny je dobré použití hodnoty inset.

Background-image – Přidání obrázku na pozadí elementu prostřednictvím vlastnosti background-image je běžnou součástí webové tvorby a poměrně oblíbenou praktikou. Dokáže uživatele na stránce snadno upoutat a oslovit ho. Může mít však negativní dopad na výkon webové stránky. Problémy nastávají zejména při použití rozsáhlých obrázků nebo při vyšším počtu obrázků, což vede ke značnému zpomalení. Pro efektivnější využití vlastnosti background-image existuje několik účinných postupů, těmi jsou zmenšení velikosti souborů obrázků, využití metod komprese obrázků, jako je optimalizace formátů JPEG nebo PNG, použití techniky image sprites, pomocí které lze z více obrázku na stránce vytvořit jeden a usnadnit tak jeho vykreslování. Dále se nabízí Implementace techniky lazy loading, díky které se obrázky načítají pouze v okamžiku, kdy je to skutečně potřeba.

Border-radius – Vlastnost border-radius je elegantním řešením pro zakomponování zaoblených rohů jednotlivým elementům. Nicméně, její nerozvážné použití může způsobit pomalejší načítání stránek, zejména když je aplikována na velké množství prvků nebo s výrazně zaoblenými rohy. Vyhnout se tomu dá pomocí nižší hodnoty border-radius, použitím vlastnosti border-image místo border-radius nebo použitím SVG grafiky.

Transform – Vlastnost transform je kouzelným nástrojem, umožňujícím oživit stránky různými transformacemi jako jsou rotace, změna měřítka, či zkosení. I zde platí, že její neopatrné použití může být pro výkon stránky značně nákladné. K zachování lehkosti a rychlosti stránky, stačí dodržet pár jednoduchým doporučení. Vždy je třeba dávat přednost 2D transformacím, které jsou méně náročné na výkon, než jejich 3D transformace. Při implementaci animací, je třeba využít vlastnost will-change, což umožní hladší a efektivnější

animace. Jako řešení se také nabízí použití funkce hardwarové akcelerace, pomocí které bude více využívána grafická karta a dojde tak k odlehčení procesoru a výsledná rychlost bude vyšší.

Filter – Vlastnost filter dodává jednotlivým elementům na webu vizuální efekty jako rozmazání, jas či úpravu barev. Pro optimalizaci se opět nabízí použití hardwarové akcelerace pomocí vlastnosti transform-style nebo použití vlastnosti will-change. [28]

Position – I když se to zprvu nemusí zdát jako jasná informace, i taková vlastnost position, která umožňuje šikovně umístit prvky kamkoli na stránce, může být použita špatně a tím může omezit rychlost naší stránky. Zejména tak absolutní nebo fixní pozicování, může rychlosti webu uškodit. Každý prvek, který je takto na stránku umístěn, nutí prohlížeč k novému výpočtu celého rozvržení stránky. Čím více výpočtů, tím pomalejší načítání. To je důvod, proč i z tak základním nástrojem pro pozicování je nutné zacházet uvážlivě.[29]

3.10 Analýza a měření výkonnosti webových stránek

Při webové tvorbě se snažíme o to, abychom maximalizovali uživatelovu „user experience“ neboli uživatelský prožitek. Ten hraje pro dosažení úspěchu celé stránky, respektive daného produktu, klíčovou roli. Je třeba si klást za cíl maximalizovat celkový pocit, požitek a spokojenost uživatele na naší webové stránce a zabezpečit si jeho opětovný návrat. Uživatelský prožitek je poměrně subjektivní záležitost a je posuzován individuálně každým uživatelem na základě jeho interakce se stránkou. Celkově lze posuzovat několika různými faktory. Takovými faktory může být například to, jak se lze na dané webové stránce přehledně navigovat v nabídce, jak snadná je orientace mezi jednotlivými prvky, kolik úsilí zabere vyhledat informaci, kterou uživatel hledá anebo v neposlední řade kolik času zabere, než se uživatel po kliknutí na odkaz na naši stránku dostane. Tato poslední vlastnosti by se dala považovat za jednu z vůbec nejdůležitějších. Rychlost, respektive doba, po kterou se webová stránka načítá, hraje klíčovou roli v uživatelově prožitku. A to z toho důvodu, že zde soupeříme o to, zda daný uživatel na stránku vůbec vstoupí nebo ji opustí ještě před tím, než se stihla načíst. Hodnota, kterou se nám v druhém případě podaří uživateli předat, pak zůstane nulová.

---

[28] LE, Duc. *Costly CSS Properties and How to Optimize Them*. *Dev* [online]. 2023 [cit. 2024-01-14]. Dostupné z: https://dev.to/leduc1901/costly-css-properties-and-how-to-optimize-them-3bmd?fbclid=IwAR27oWGA7oFQz7ko8-mFsyiDwF9piaC-IKw6dutBfNAEgEesbtxhsmA-Eq0

[29] KARIMOV, Farda. *CSS Optimization: Avoiding and Optimizing for Faster Website Speed*. Medium [online]. 2023. [cit. 2024-01-14]. Dostupné z: https://levelup.gitconnected.com/css-optimization-avoiding-and-optimizing-for-faster-website -speed-ccd33854cd01.

A právě proto je nezbytné se zajímat o rychlost načítání webové stránky a snažit se celý proces co nejvíce zoptimalizovat a uživatelský prožitek zlepšovat a adaptovat ho dle aktuálních uživatelských požadavků. [30]

Uživatelé očekávají okamžitý přístup k informacím a rychlou interakci webové stránky. Proto rychlost načítání stránky je klíčovou determinantou uživatelského prožitku. Jakmile načítání webové stránky trvá déle jak 1 sekundu, uživatelé už pociťují zpoždění. Pokud se webová stránka načítá rychle, uživatel ji vnímá jako responzivní, přívětivou a je velká pravděpodobnost, že se opětovně vrátí.

Je důležité brát v potaz i následující. Rychlost, jak se jednomu uživateli webová stránky načítá, nemusí vůbec souviset s rychlostí načítání stránky uživateli druhému. Webová stránka se může na rychlém zařízení a s rychlým připojením na internet načítat rychle, a naopak na pomalém zařízení s pomalým internetem načítat pomalu. Dále je potřeba brát v potaz, v jaké fázi a co přesně chceme měřit. Jak bylo na začátku této práce uvedeno, nejdříve uživatel reaguje s prohlížečem, ten pak reaguje se serverem, ten následně posílá požadavek dál zpět přes prohlížeč na uživatele, mezitím probíhá komunikace například s DNS serverem a mnoho dalšího. Je třeba si uvědomit, že toto všechno vstupuje do času načítání. Proto je nesmírně důležité určit si v jaké fázi rychlost chceme měřit a následně jakékoli měření objektivně vztáhnout k určitým kritériím a sledovat rychlost za stejných nebo podobných podmínek. Pouze tak bude mít měření vypovídající hodnotu.[31]

Měřící metriky můžeme rozdělit do dvou skupin podle toto, kde a jak je samotné měření prováděno. První skupinou měření je měření v takzvaném „kontrolovaném prostředí“. Toto měření se postará o sjednocení veškerých faktorů, jako je rychlost internetového připojení, odlišný výkon zařízení nebo ostatní věci jako jsou například narušující prvky reklam, které mohou mít signifikantní vliv na rychlost načítání. Měření v kontrolovaném prostředí nám tak nabízí komplexní analýzu rychlosti, oproštěnou od veškerých nepřesných nuancí.

Oproti tomu měření v takzvaném „reálném prostředí“, nám tak přesné výsledky nezaručí a nebude nám reflektovat, jak se bude webová stránka načítat širokému spektru ostatních uživatelů. Rychlost v tomto případě může být ovlivněna nestabilním internetovým připojením

---

[30] EDGAR, Matthew. *Speed Metrics Guide* [online]. Apress Berkeley, CA, 2024 [cit. 2024-03-09]. Dostupné z: doi: https://doi.org/10.1007/979-8-8688-0155-6

[31] WALTON, Philip. *User-centric performance metrics*. Web.dev [online]. 2019 [cit. 2024-01-14]. Dostupné z: https://web.dev/art icles/user-centric-performancemetrics?fbclid=IwAR2f8lkc3CxPLvSYVo29tiaXwPw7cgvi lpwKJCK7z1ucfgXF vkfJIIFi9ko#important-metrics-to-measure.

nebo třeba měnícími se reklamami. Při měření načítání webové stránky se můžeme zaměřit na několik oblastí, které souvisí s tím, jak uživatelé vnímají její výkon. Těmi jsou:

- **Vnímaná rychlost načítání**: Rychlost, s jakou se stránka může načíst a zobrazit všechny vizuální prvky na obrazovce.
- **Odezva načítání**: Rychlost načtení a exekuce Java Scriptového kódu, který umožnuje jednotlivým prvkům reagovat na uživatelovu interakci.
- **Odezva za běhu**: Jak rychle dokáže webová stránka, která už je načtena, reagovat na uživatelovu interakci.
- **Vizuální stabilita**: Kontrola, zda se prvky na stránce nepohybují neočekávaně, nesprávným směrem, nebo nezpůsobují kolizi s ostatními prvky.
- **Hladkost načítání**: Vykreslování prvků, jejich animace, plynulé a konzistentní přecházení z jednoho stavu do druhého.[32]

3.11 Web Vitals

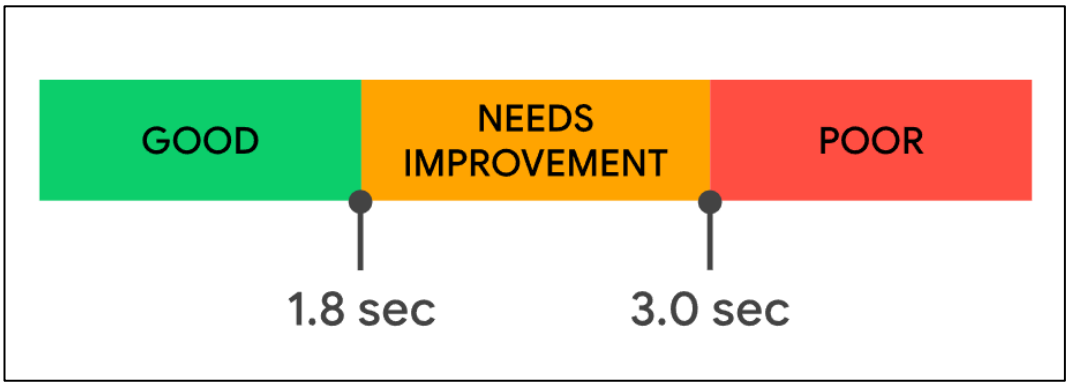
Core Web Vitals je sada metrik od Google, poskytující základní měřitelné hodnoty pro kvalitní uživatelskou zkušenost. Dá se hovořit o nastaveném benchmarku pro webové stránky, které jsou univerzálně použitelné. Zaměřují se na načítání, interakci a vizuální stabilitu se specifickými metrikami, které budou detailně vysvětleny v následujícím textu. Aktuální, takzvanou stabilní sadou Core Web Vitals jsou tři ukazatele. Těmi jsou LCP, FID a CLS. Kromě těchto tří je zde ještě metrika INP, která je ve fázi takzvaných „pending metrics“ tedy budoucí Core Web Vitals, která by se měla implementovat začátkem roku 2024. Core Web Vitals jsou důležité pro SEO, protože přímo ovlivňují uživatelskou zkušenost (UX) na webu, což ve výsledků může ovlivnit jejich postavení ve vyhledávačích. Zatímco Core Web Vitals splňující svůj požadovaný interval a jsou kritickými pro poskytování dobré UX, existují i další důležité metriky označované jako Web Vitals. Web Vitals slouží jako zástupní nebo doplňkové metriky pro Core Web Vitals, aby pomohly zachytit jinou část nebo detail při diagnostice konkrétního problému.[33]

---

[32] WALTON, Philip. *User-centric performance metrics*. Web.dev [online]. 2019 [cit. 2024-01-14]. Dostupné z: https://web.dev/art icles/user-centric-performancemetrics?fbclid=IwAR2f8lkc3CxPLvSYVo29tiaXwPw7cg vilpwKJCK7z1ucfgXF vkfJIIFi9ko#important-metrics-to-measure.

[33] EDGAR, Matthew. *Speed Metrics Guide* [online]. Apress Berkeley, CA, 2024 [cit. 2024-03-09]. Dostupné z: doi: https://doi.org/10.1007/979-8-8688-0155-6

First Contentful Paint (FCP) – FCP je ukazatel, který monitoruje dobu mezi počátečním uživatelovým přechodem na webovou stránku a prvním výskytem jakéhokoli obsahu, který se objeví na jeho obrazovce. Tento obsah zahrnuje text, obrázky, svg prvky a ostatní nebílé prvky. Je důležité rozlišovat mezi FCP, který zachycuje počáteční vykreslování a LCP (Largest Contentful Paint), který označuje to, když už se většina obsahu plně načetla. Pro optimální UX by se webové stránky měly zaměřit na FCP 1,8 sekundy nebo méně, přičemž dobrým posouzením je 75. percentil načítacího času napříč různými zařízeními. Je důležité si uvědomit, že FCP je ovlivněno několika faktory, jako je doba potřebná k navázání připojení, doba uvolnění (unload time) z předchozí stránky, doba přesměrování nebo TTFB. [34]



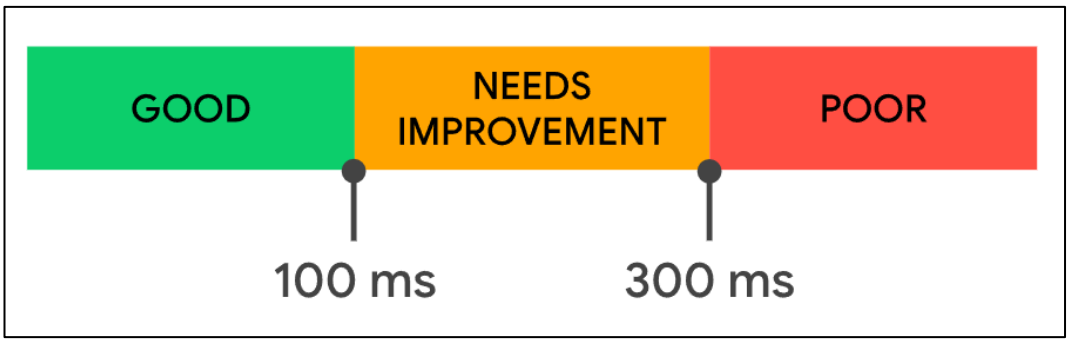
Obr. č. 5: Hodnocení metriky FCP[35]

Largest Contentful Paint (LCP) – LCP aneb největší vykreslení obsahu, je metrika, která zaznamenává dobu vykreslení největšího bloku textu nebo obrázku v zorném poli uživatele od okamžiku, kdy stránku poprvé otevřel. Aby web poskytoval vynikající UX, měl by cílit na hranici LCP nepřesahující 2,5 sekundy. Jako spolehlivý ukazatel pro sledování tohoto cíle slouží opět 75. percentil časů načítání stránek, rozdělený podle mobilních a stolních zařízení. I zde je třeba brát na zřetel, že do LCP vstupují i ostatní faktory jako je přesměrování a Time To First Byte (TTFB), které mohou mít významný vliv.[36]

[34] WALTON, Philip. *First Contentful Paint (FCP)*. Web.dev [online]. 2023 [cit. 2024-01-14]. Dostupné z: https://web.dev/articl es/fcp.
[35] WALTON, Philip. *First Contentful Paint (FCP)*. Web.dev [online]. 2023 [cit. 2024-01-14]. Dostupné z: https://web.dev/articl es/fcp.
[36] WALTON, Philip a POLLARD, Barry. *Largest Contentful Paint (LCP)*. Web.dev [online]. 2023 [cit. 2024-01-14]. Dostupné z: https://web.dev/articles/lcp.

Obr. č. 6: Hodnocení metriky LCP[37]

First Input Delay (FID) – Porozumění a neustálé zlepšení UX na webových stránkách je klíčové. UX závisí na mnoha faktorech, včetně designu a responzivity. Zatímco posouzení designu a struktury je poměrně komplikované, měření odezvy nikoli, a to díky metrikám, jako je First Contentful Paint (FCP) a First Input Delay (FID). FCP sleduje dobu zobrazení obsahu, ale FID se ponoří ještě hlouběji a sleduje zpoždění od první interakce uživatele na daný prvek až do reakce prohlížeče. Cílem je hodnota FID 100 milisekund nebo méně. Toto zaměření na rychlost a interaktivitu pomáhá zajistit, aby uživatelé byli již od první interakce na stránce spokojeni.[38]

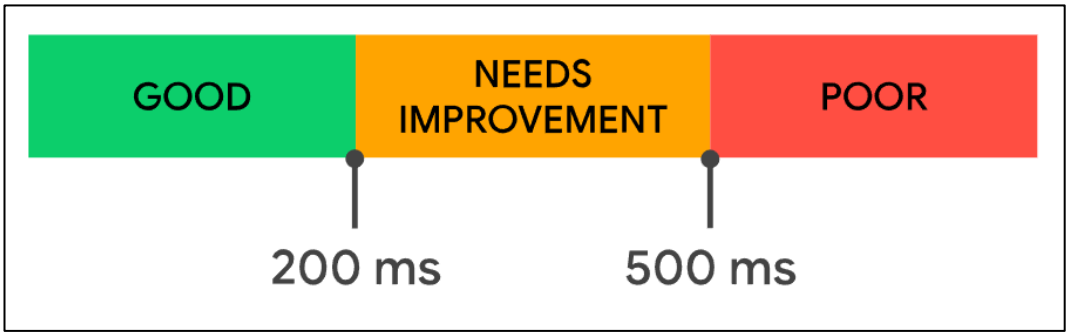


Obr. č. 7: Hodnocení metriky FID[39]

Interaction to Next Paint (INP) – Metrika Interaction to Next Paint (INP) je nadcházející Core Web Vital, která by měla být platná v roce 2024. Metrika je navržena tak, aby nahradila ne příliš objektivní FID ukazatel měřením odezvy stránky. Oproti FID zachycuje

[37] WALTON, Philip a POLLARD, Barry. *Largest Contentful Paint (LCP)*. Web.dev [online]. 2023 [cit. 2024-01-14]. Dostupné z: https://web.dev/articles/lcp.
[38] WALTON, Philip. *First Input Delay (FID)*. Web.dev [online]. 2023 [cit. 2024-01-14]. Dostupné z: https://web.dev/articles/fid.
[39] WALTON, Philip. *First Input Delay (FID)*. Web.dev [online]. 2023. [cit. 2024-01-14]. Dostupné z: https://web.dev/articles/fid.

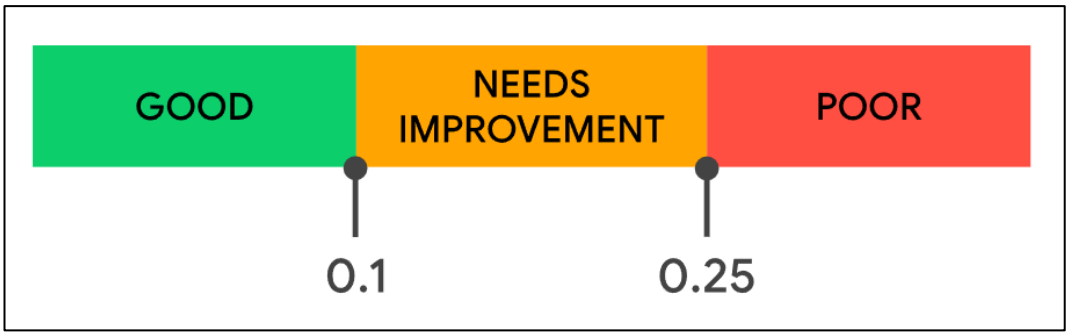
latenci všech uživatelských interakcí a zaměřuje se na nejhorší případ, tedy nejdelší zpožděnou reakci. INP je počítáno na základě nejdelší interakce, při vyřazení extrémů, a tak nabízí podrobnější pochopení interaktivního výkonu stránky s cílem udržet krátké doby odezvy pro většinu uživatelských interakcí a ne pouze pro počáteční. Dobré skóre INP je obvykle 200 milisekund nebo méně, což odráží rychlé a citlivé UX.[40]



Obr. č. 8: Hodnocení metriky INP[41]

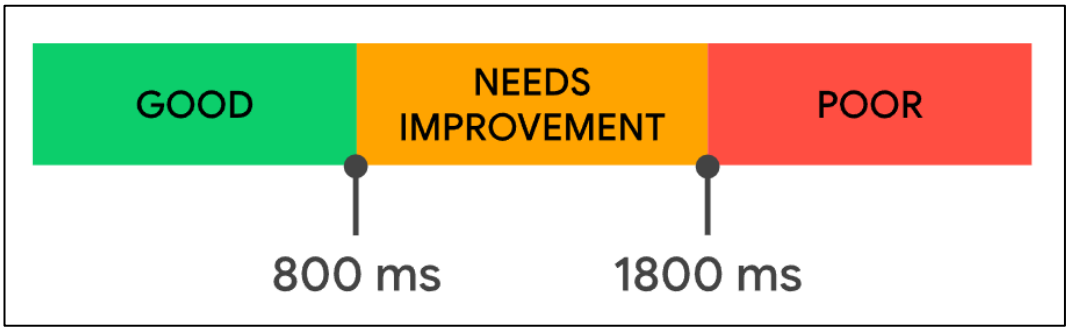
Time to Interactive (TTI) – TTI udávají, kdy se stránka stává plně interaktivní. V tomto se odlišuje od TBT, které kvantifikuje zpoždění interaktivity. TTI měří čas od začátku prvního vykreslení obsahu až po čas, kdy může konzistentně reagovat na vstup uživatele. TBT se na druhou stranu zaměřuje na součet jednotlivých zpoždění trvajících déle než 50 milisekund, kdy bylo hlavní vlákno blokováno pro další interaktivitu. TBT v podstatě přispívá k detailnímu pochopení zpoždění, která uživatelé na stránce zažívají.[42]

[40] POLLARD, Barry a WAGNER, Jeremy. *Interaction to Next Paint.* Web.dev [online]. 2023 [cit. 2024-01-14]. Dostupné z: https://web.dev/articles/inp.
[41] POLLARD, Barry a WAGNER, Jeremy. *Interaction to Next Paint.* Web.dev [online]. 2023 [cit. 2024-01-14]. Dostupné z: https://web.dev/articles/inp.
[42] WALTON, Philip. *Time to Interactive (TTI)*. Web.dev [online]. 2023 [cit. 2024-01-14]. Dostupné z: https://web.dev/articles/tti.



Obr. č. 9: Hodnocení metriky TTI[43]

TTFB – TTFB neboli "Time to First Byte" (Čas k prvnímu bytu), je ukazatel, který sleduje dobu od zaslání požadavku po okamžik, kdy dorazí první byte odpovědi. Tento ukazatel je výsledkem sumy několika fází. Od času potřebného pro přesměrování, přes dobu vyhledávání DNS, proces navazování spojení a vyjednávání TLS, až po okamžik, kdy dorazí první byte dat. Vzhledem k tomu, že Time to First Byte (TTFB) předchází metrikám zaměřených přímo na UX, jako je First Contentful Paint (FCP) a Largest Contentful Paint (LCP), je vhodné, aby servery reagovaly rychle a posílaly požadavky neprodleně. Jako obecný cíl se webům doporučuje dosáhnout TTFB 0,8 sekundy nebo rychlejší. Důležité je zabezpečit tento čas takovým způsobem, aby metriky, které jsou na TTFB přímo závislé, dokázaly splnit 75 percentil načítání. I přesto, že TTFB není Core Web Vitals metrikou a splnění prahu „dobré“ pro TTFB není striktně vyžadováno, je důležité, aby weby zajistily, že to nebude bránit jejich výkonu v jiných klíčových metrikách.[44]



Obr. č. 10: Hodnocení metriky TTFB[45]

[43] WALTON, Philip. *Time to Interactive (TTI)*. Web.dev [online]. 2023 [cit. 2024-01-14]. Dostupné z: https://web.dev/articles/tti.
[44] WAGNER, Jeremy a POLLARD, Barry. *Time to First Byte (TTFB)*. Web.dev [online]. 2023 [cit. 2024-01-14]. Dostupné z: https://web.dev/articles/ttfb.
[45] WAGNER, Jeremy a POLLARD, Barry. *Time to First Byte (TTFB)*.Web.dev [online]. 2023 [cit. 2024-01-14]. Dostupné z: https://web.dev/articles/ttfb.

Total Blocking Time (TBT) – TBT neboli celková doba blokování je ukazatel kvantifikující dobu, po kterou je stránka neinteraktivní, než se stane plně interaktivní. Představuje celkovou dobu trvání, po kterou je hlavní vlákno načítání stránky blokováno úkoly po dobu více než 50 milisekund po prvním vykreslení obsahu (FCP). Optimální TBT je méně než 200 milisekund pro UX zajišťující, že web bude plně responzivní a dobře použitelný.[46]

Cumulative Layout Shift (CLS) – CLS je klíčová Core Web Vitals metrika měřící vizuální stabilitu sledováním neočekávaných posunů prvků na stránce. Měří se podle posunu v rámci okna relace, což ukazuje, kolik viditelného obsahu se posunulo a jak daleko. Výpočet zahrnuje podíl dopadu (dotčená oblast výřezu) a podíl vzdálenosti (pohyb prvku vzhledem k výřezu). Za dobré skóre je považování skóre CLS 0,1 nebo nižší, což představuje minimální pohyb obsahu pro optimální uživatelský zážitek. Toto skóre pomáhá vývojářům porozumět a zlepšit stabilitu a konzistenci.[47]

[46] WALTON, Philip. *Total Blocking Time (TBT)*. Web.dev [online]. 2023 [cit. 2024-01-14]. Dostupné z: https://web.dev/articles/tbt.
[47] MIHAJLIJA, Milica a WALTON, Philip. *Cumulative Layout Shift (CLS)*. Web.dev [online]. 2023 [cit. 2024-01-14]. Dostupné z: https://web.dev/articles/cls.

4 Vlastní práce

Praktická část této bakalářské práce se zaměřuje na detailní analýzu vlivu různých vlastností a nástrojů CSS na rychlost načítání webové stránky. Klade důraz na identifikaci, jak rozdílné techniky a přístupy při implementaci CSS ovlivňují efektivitu a rychlost načítání webového obsahu. Předmětem zkoumání nejsou pouze samotné CSS vlastnosti, význam je přikládán i CSS frameworkům nebo struktuře psaní a importu CSS souboru. Následně byla velká pozornost zaměřena na kvantifikaci těchto přístupů a technik, jejich porovnání mezi sebou a vyvození vhodných doporučení a postupů. Veškerá kvantifikace proběhla za pomocí způsobů a metrik zmíněných v teoretické části práce.

Pro účely těchto analýz bylo vytvořeno několik vzorových testovacích webových stránek. Tyto stránky se lišily svým vzhledem a náročností, dle požadavku na testované nástroje a vlastnosti. Veškeré vzorové testovací webové stránky byly vytvořeny ve vývojovém prostředí Microsoft Visual Studio s využitím značkovacího jazyka HTML a kaskádových stylů CSS. Při testování a analýze vlivu použití frameworků CSS byly také použitý CSS frameworky Bootstrap a Tailwind. Celá výzkumná část byla zaměřena na několik klíčových aspektů jazyka CSS jako je testování layoutových modelů, načítání stylů z jednoho či více souborů, testování drahých vlastností CSS a porovnání CSS frameworků. Úmyslně se předešlo použití JavaScriptu, aby bylo možné izolovat vliv na rychlost načítání pouze na samotné CSS a nehrozilo ovlivnění jinými skutečnostmi.

Všechna měření byla provedena za konstantních podmínek, aby se zabezpečilo vypovídajícím schopnostem naměřených výsledků. Všechny HTML značky a CSS styly byly psány a aplikovány v souladu s dodržováním pravidel pro tyto jazyky. Měření byla vykonávána na zařízení Lenovo IdeaPad Gaming 3 s následujícími vlastnostmi: GPU: NVIDIA GeForce GTX 1650 a CPU: AMD Ryzen 5 4600 H (Radeon Graphics), RAM: 8 GB, při daných parametrech internetového připojení: rychlost stahování 50Mb/s a 10Mb/s pro rychlost nahrávání. Každé měření bylo opakováno desetkrát a poté zaznamenáno do předem připravené excelovské tabulky. Následně byla provedena kontrola, zda se nějaké měření výrazně neodchyluje od naměřených průměrných hodnot. Pokud došlo v některém z měření k extrémnímu vychýlení hodnot, byla hodnota odstraněna a měření opakováno. Naměřené hodnoty byly v závěrečné fázi vyhodnoceny, přepracovány do grafických a tabulkových podob a detailně okomentovány.

Pro měření vlivu různých vlastností a nástrojů CSS na rychlost načítání byly použity nástroje PageSpeed Insight a Chrome DevTools. V iniciální fázi analýzy byl použit nástroj

Chrome DevTools, který je zabudovaný přímo jako součást prohlížeče Google Chrome. Tento nástroj umožňuje detailní sledování časů načítání a vizualizaci zátěže jednotlivých zdrojů v reálném čase. Následně byl pro rozšířenou a globální analýzu použit online nástroj PageSpeed Insight od společnosti Google. Tento nástroj nabízí konkrétní přehledy s jednotlivými metrikami pro měření výkonu a optimalizace stránek doplněné o jednotlivá doporučení pro zlepšení. Spojením těchto přístupů bylo možno získat komplexní a detailní pohled na faktory ovlivňující rychlost načítání.

4.1 Layoutové techniky CSS a rychlost načítání

Webový layout představuje rozložení a uspořádání viditelných prvků na stránce. Optimálním rozmístěním těchto prvků mezi sebou, jejich hladkou návazností a správným pořadím získáme možnost kontrolovat uživatelský prožitek a také uživatele na stránce efektivně směřovat kam chceme. Správně vytvořený layout dokáže přilákat uživatelovu pozornost na námi zvolené místo. [48] Z technického hlediska může správně navržený layout minimalizovat množství potřebných dat ke stažení a zpracování prohlížečem, což má přímý dopad na dobu načítání webové stránky. Vhodné řešení layoutu, které bere v úvahu nejen estetické, ale i výkonnostní aspekty, je proto nezbytné pro optimalizaci rychlosti načítání. To zahrnuje volbu mezi různými layoutovými technikami, jako jsou float, flexbox a grid, avšak každá z nich může mít odlišný vliv na rychlost načítání. Proto se tato kapitola zaměřuje na detailní analýzu těchto technik, s cílem poskytnout ucelený pohled na to, jak správně navržený layout může přispět k lepší rychlosti načítání, a tím i k celkově lepšímu uživatelskému prožitku.[49]

Pro účely této práce byly vytvořeny celkem tři testovací webové stránky, každá z nich využívající jednu z výše zmíněných layoutových technik: float, flexbox a grid. Na každé vytvořené stránce pro konkrétní testovanou vlastnost se vždy měnil pouze počet prvků. Konkrétně bylo pracováno s počtem 4, 20, 40, 80, 160 a 300 prvků. Každá stránka byla navržena tak, aby byla co nejpodobnější ostatním ve smyslu obsahu a funkcí a lišila se pouze v použité layoutové technice. Cílem bylo také minimalizovat vliv ostatních faktorů na rychlost načítání a zaměřit se specificky na vliv layoutových technik. K měření rychlosti načítání byl

[48] SENDPULSE. *Website Layout*. Sendpulse [online]. 2023 [cit. 2024-02-27] Dostupné z: https://sendpulse.com/support/glossary/websi te-layout.
[49] THEWEBSITEARCHITECT. *Why is a layout important?* Thewebsitearchitect [online]. 2020 [cit. 2024-02-27]. Dostupné z: https://the websitearchitect.com/why-is-the-layout-of-a-website-important/.

použit nástroj Chrome DevTools, který poskytují komplexní přehled statistik z hlediska načítání webové stránky.

Aby bylo zabezpečeno konzistentní prostředí, byl před samotnou analýzou jednotlivých technik zhotoven základ stejnorodého kódu HTML a CSS, který se shodoval napříč všemi stránkami v této kapitole. Následující obrázky názorně zobrazují tyto základní kódy, od kterých byly všechny stránky odvozeny.

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="stylesheet" href="styles.css" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Layoutové techniky</title>
  </head>
  <body>
    <div class="stranka">
      <div class="wrapper">
        <div class="obsah"><p></p></div>
        <div class="obsah"><p></p></div>
        <div class="obsah"><p></p></div>
        <div class="obsah"><p></p></div>
      </div>
    </div>
  </body>
</html>
```

Obr. č. 11: Základní HTML kód (zdroj: autor práce)

První obrázek (č. 11) představuje HTML kód, který tvořil základní strukturu všech stránek. Následující obrázek (č. 12) představuje použité kaskádové styly.

```css
body {
  padding-left: 200px;
  padding-right: 200px;
  background-color: #e6e6e6;
}

.stranka {
  background-color: white;
  height: 11000px;
  padding-left: 30px;
  padding-right: 30px;
  padding-top: 20px;
}

p {
  font-size: 25px;
  font-family: sans-serif;
  text-align: center;
}
```

Obr. č. 12: Základní CSS kód (zdroj: autor práce)

Po vytvoření základního kódu následovala tvorba stránek pro jednotlivé testované CSS vlastnosti. Celkem tímto způsobem vznikly tři testovací webové stránky, první pro vlastnost float, další pro vlastnost flex a poslední pro vlastnost grid. Do každé takto vytvořené testovací webové stránky bylo pro účely měření vlivu a náročnosti přidáváno určité množství stejných HTML elementů. Postupně bylo tímto způsobem do každé testovací stránky přidáváno 4, 20, 40, 80, 160 až 300 elementů. V každém kroku bylo provedeno 10 měření, ze kterých byly postupně zaznamenávány hodnoty renderingu, paintingu a celkové doby vykreslení. Tyto hodnoty byly následně zprůměrovány.

4.1.1 Float

Vlastnost float se využívá pro pozicování a formátování obsahu. Využitím této vlastnosti je prvek odebrán z běžného toku prvků stránky. Nadále je však jeho součástí, což není pravdou například u absolutního pozicování. Vlastnost float slouží k umístění elementu na levou nebo pravou stranu jeho nadřazeného elementu a umožňuje tak provést formátování.[50]

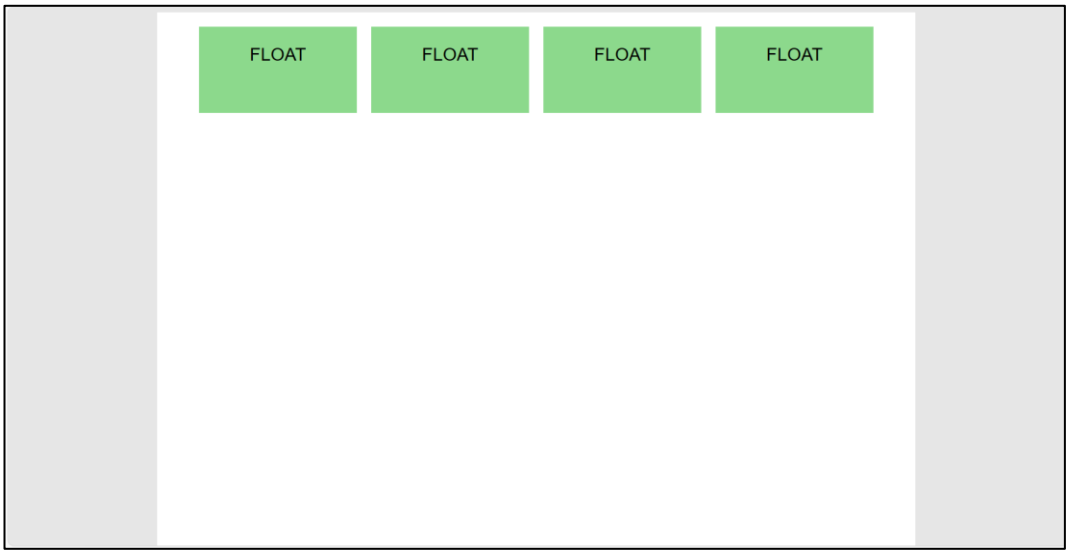
Jako první byla provedena analýza této CSS vlastnosti float. Na následujícím obrázku je možné vidět strukturu CSS kódu, jakým byla vlastnost float do stránky implementována. Na obrázku č. 13 je zobrazen konkrétní CSS kód.

```
.wrapper {
  width: 960px;
  margin-left: auto;
  margin-right: auto;
}

.wrapper:after {
  content: "";
  display: block;
  clear: both;
}

.obsah {
  float: left;
  width: 220px;
  height: 120px;
  background-color: #8cd98c;
  margin: 0 10px 15px 10px;
}
```

Obr. č. 13: CSS kód pro testovanou vlastnost float (zdroj: autor práce)

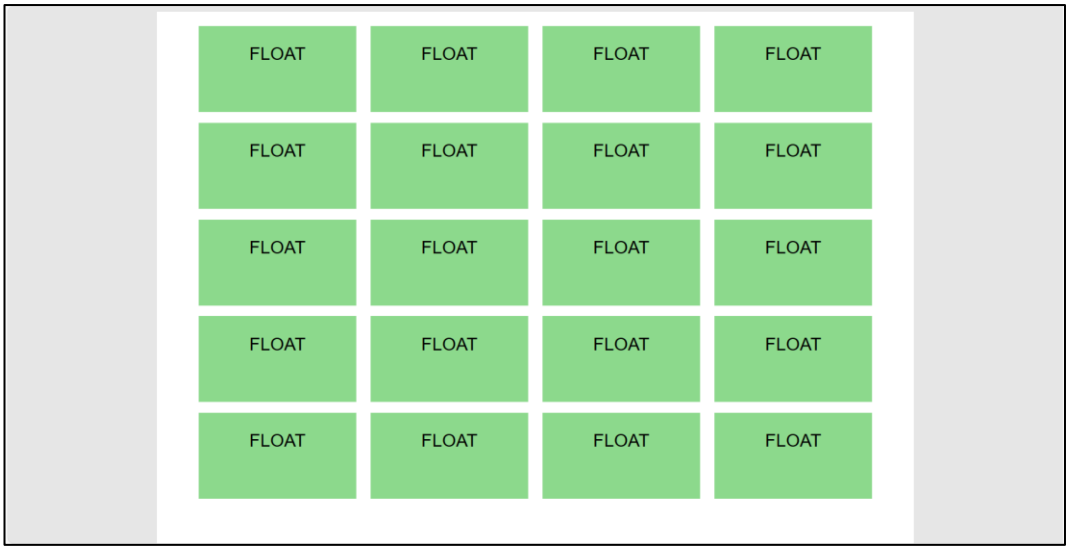
[50] MDN Web Docs. *Float*. MDN Web Docs [online]. 2024 [cit. 2024-02-27]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/CSS/float.

Na následujících obrázcích je vidět výsledná vizuální podoba testovaných HTML prvků a jejich zvyšujícího se množství.



Obr. č. 14: CSS kód vykreslen prohlížečem (zdroj: autor práce)

Pro zachování přehlednosti práce byla zvolena ukázku pouze prvního a druhého případu, tedy pro 4 prvky (obrázek č. 14) a 20 prvků (obrázek č. 15). Analogickým způsobem byly přidány, změřeny a vyhodnoceny všechny případy množství prvků.



Obr. č. 15: CSS kód vykreslen prohlížečem (zdroj: autor práce)

Ve všech kategoriích testovaného množství prvků a pro všechna opakující se měření bylo pro vlastnost float zpracováno celkem 60 měření (10 pro konkrétní množství). Naměřené

průměrné hodnoty pro konkrétní množství měřených prvků, rozdělených do dvou sloupců rendering a painting jsou zpracovány v následující tabulce:

POČET HTML PRVKŮ	RENDERING [ms]	PAINTING [ms]	CELKEM [ms]
4	2,82	1,47	4,29
20	3,99	2,13	6,12
40	4,56	2,34	6,90
80	6,96	3,91	10,87
160	10,20	5,55	15,75
300	17,16	5,70	22,86

Tab. č. 1: Výsledky testované vlastnosti float (zdroj: autor práce)

4.1.2 Flexbox

Flexbox je moderní layoutová technika pro vytváření jednorozměrného rozvržení prvků. V praxi to tedy znamená, že pomocí něho můžeme zarovnat a pozicovat prvky v řádku nebo ve sloupci. Flexbox usnadňuje vytváření komplexních layoutů se zachováním jednoduchého kódování, aniž by bylo nutné používat tradiční techniky, jako je již dříve zmíněný float.[51]

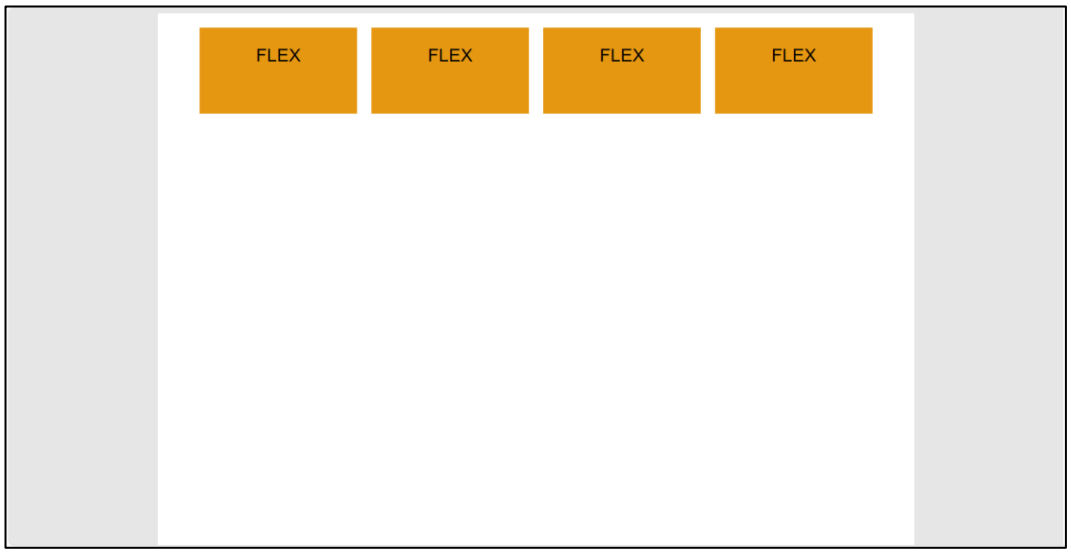
Jako druhá byla provedená analýza výše zmíněné vlastnosti flexbox. Stejně jako v předchozím testování vlastnosti, i v tomto případě byla stránka tvořena výše uvedeným základním kódem, do kterého byla přidána CSS vlastnost flex. Struktura CSS kódu, který byl testován, je vidět na následujícím obrázku. Dále je pak možné vidět vizuální podobu po spuštění stránky prohlížečem. Pro zanechání přehlednosti práce je opět uveden jako příklad počet testovaných HTML prvků 4 a 20. Další přidání prvků bylo analogické.

[51] COYIER, Chris. *A Complete Guide to Flexbox*. Css-tricks [online]. 2022 [cit. 2024-03-09]. Dostupné z: https://css-tricks.com/ snippets/css/a-guide-to-flexbox/#aa-background.

```css
.wrapper {
  display: flex;
  column-gap: 20px;
  margin-bottom: 15px;
  justify-content: center;
}

.obsah {
  width: 220px;
  height: 120px;
  background-color: #d89d36;
  text-align: center;
  font-family: sans-serif;
}
```

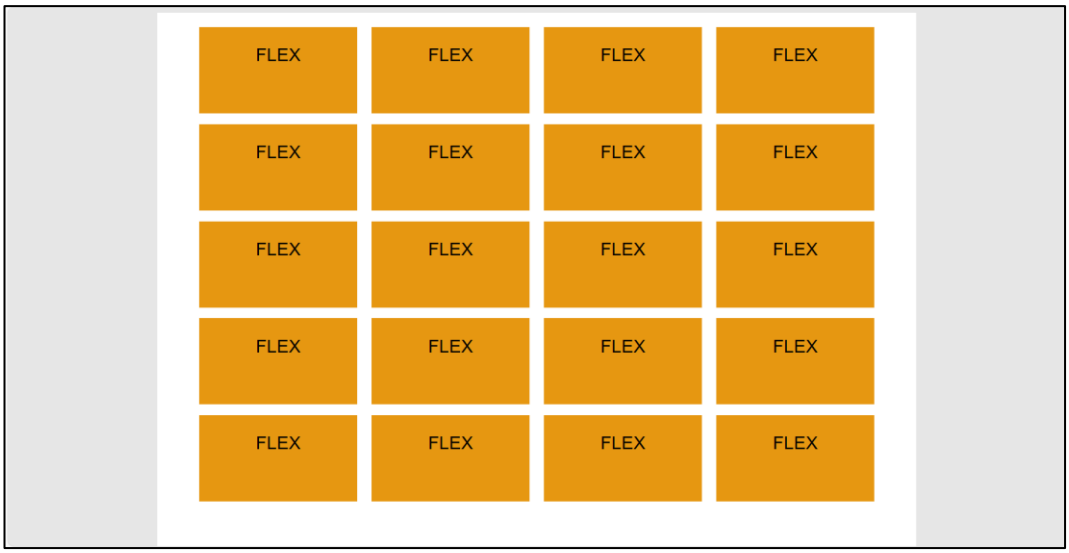
Obr. č. 16: CSS kód pro testovanou vlastnost flex (zdroj: autor práce)

Na obrázku č. 16 je možné vidět specifický CSS kód pro danou testovanou vlastnost flexbox.



Obr. č. 17: CSS kód vykreslen prohlížečem (zdroj: autor práce)

Obrázek č. 17 zobrazuje testovanou vlastnost flexbox pro 4 prvky na stránce. Na obrázku č. 18 je testovaná vlastnost flexbox pro 20 prvků.



Obr. č. 18: CSS kód vykreslen prohlížečem (zdroj: autor práce)

Na testované CSS vlastnosti flex bylo provedeno celkem 60 měření (10 pro uvedený počet prvků). Výsledné hodnoty vyplývající z analýzy je možné vidět v následující tabulce č. 2.

POČET HTML PRVKŮ	RENDERING [ms]	PAINTING [ms]	CELKEM [ms]
4	2,79	1,49	4,28
20	3,87	2,11	5,98
40	4,78	2,41	7,19
80	7,05	4,12	11,17
160	12,6	5,78	18,38
300	19,3	5,93	25,23

Tab. č. 2: Výsledky testované vlastnosti flexbox (zdroj: autor práce)

4.1.3 Grid

CSS vlastnost grid je druhá, moderní, hojně používaná pozicovací technika. Oproti flexboxu se odlišuje tím, že je dvojrozměrná. Lze pomocí ní formátovat vícerozměrné rozložení, tedy zároveň v řádku tak ve sloupci. Její síla opět spočívá ve velmi intuitivním způsobu aplikace a možné tvorby komplikovaných layoutů, kterých by bylo jinými způsoby velmi složité dosáhnout.[52] Jako poslední byla tedy testovaná CSS vlastnost grid. Vlastnost grid se odlišovala od výše zmíněných měření opět CSS kódem, který byl uzpůsoben pro tuto

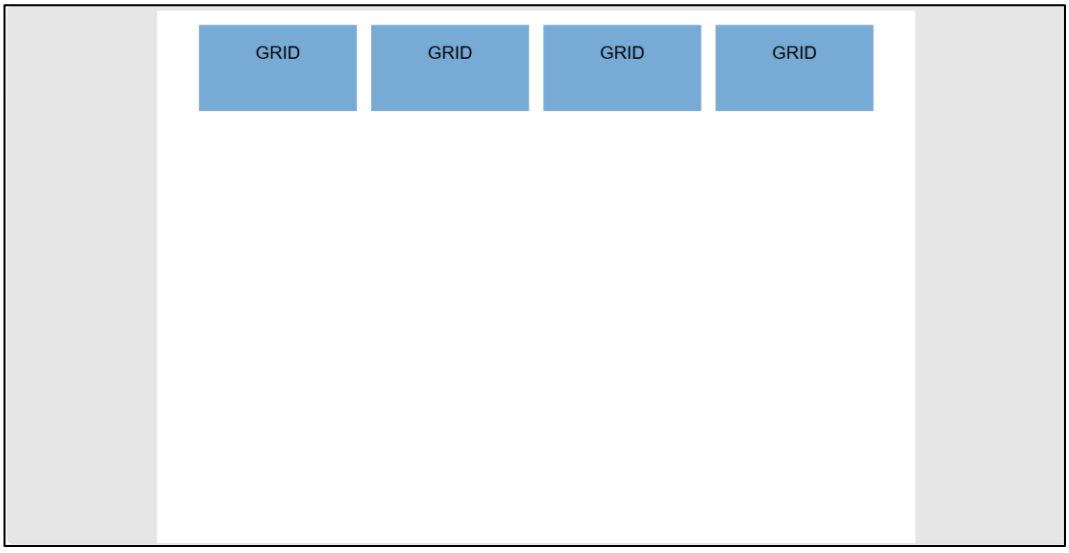
[52] MICHÁLEK, Martin. In: *CSS: moderní layout*. [Praha]: Martin Michálek – Vzhůru dolů, [2022], s. 155-167. [cit. 2024-02-27]. ISBN 978-80-88253-07-5.

konkrétní vlastnost. Kód je možné vidět na následujícím obrázku, stejně tak jako vizuální reprezentaci. Pro zachování přehlednosti byly opět pro ukázku vybrány pouze počty 4 a 20.

```
.wrapper {
  display: grid;
  grid-template-columns: 220px 220px 220px 220px;
  grid-template-rows: 120px 120px;
  column-gap: 20px;
  row-gap: 15px;
  justify-content: center;
}

.obsah {
  background-color: #649bd3;
  text-align: center;
  font-family: sans-serif;
  border-radius: 0px;
}
```

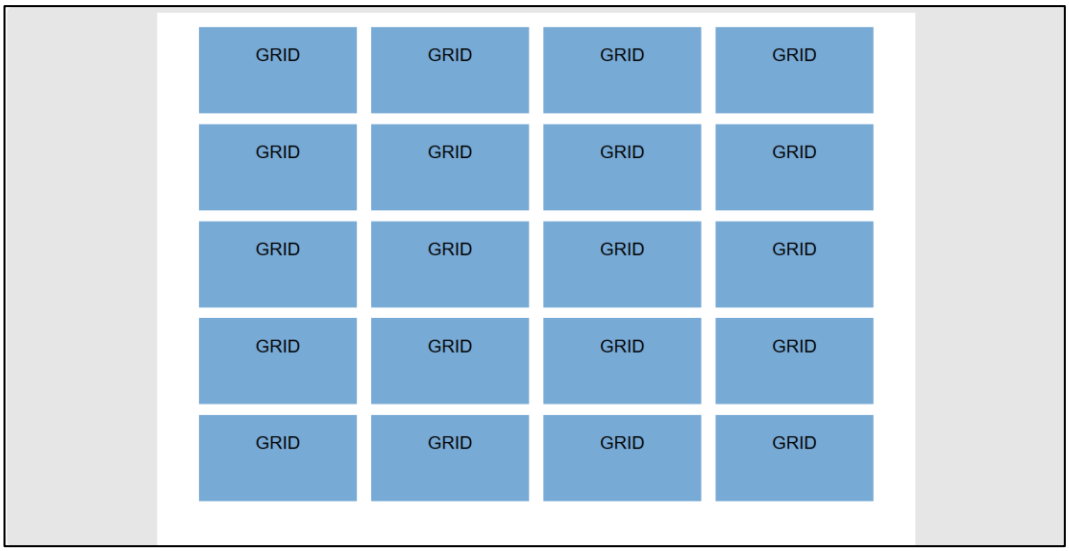
Obr. č. 19: CSS kód pro testovanou vlastnost grid (zdroj: autor práce)

Na obrázku č. 19 je možné vidět specifický CSS kód pro danou testovanou vlastnost grid.



Obr. č. 20: CSS kód vykreslen prohlížečem (zdroj: autor práce)

Obrázek č. 20 zobrazuje testovanou vlastnost grid pro 4 prvky na stránce. Na obrázku č. 21 je testovaná vlastnost grid pro 20 prvků.



Obr. č. 21: CSS kód vykreslen prohlížečem (zdroj: autor práce)

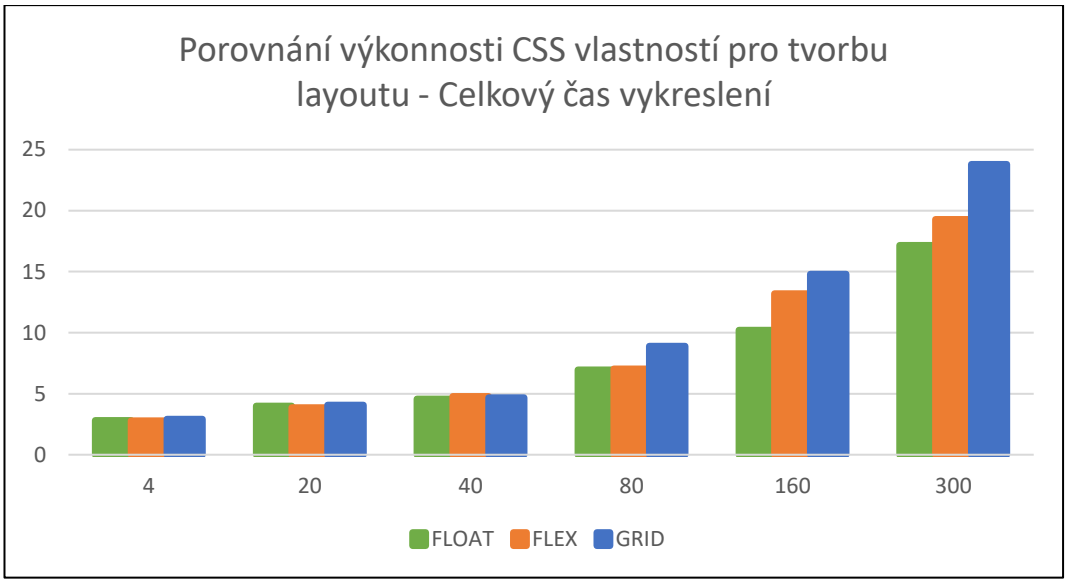
Testování CSS vlastnosti grid proběhlo stejným způsobem jako u předešlých vlastností. Měření proběhlo pro každé testované množství (od 4 do 300 prvků) po deseti opakováních. Zprůměrované naměřené výsledky jsou viditelné v následující tabulce č. 3.

POČET HTML PRVKŮ	RENDERING [ms]	PAINTING [ms]	CELKEM [ms]
4	2,93	1,51	4,44
20	4,08	2,18	6,26
40	4,68	2,39	7,07
80	8,91	4,46	13,37
160	14,8	5,88	20,68
300	23,8	6,12	29,92

Tab. č. 3: Výsledky testované vlastnosti grid (zdroj: autor práce)

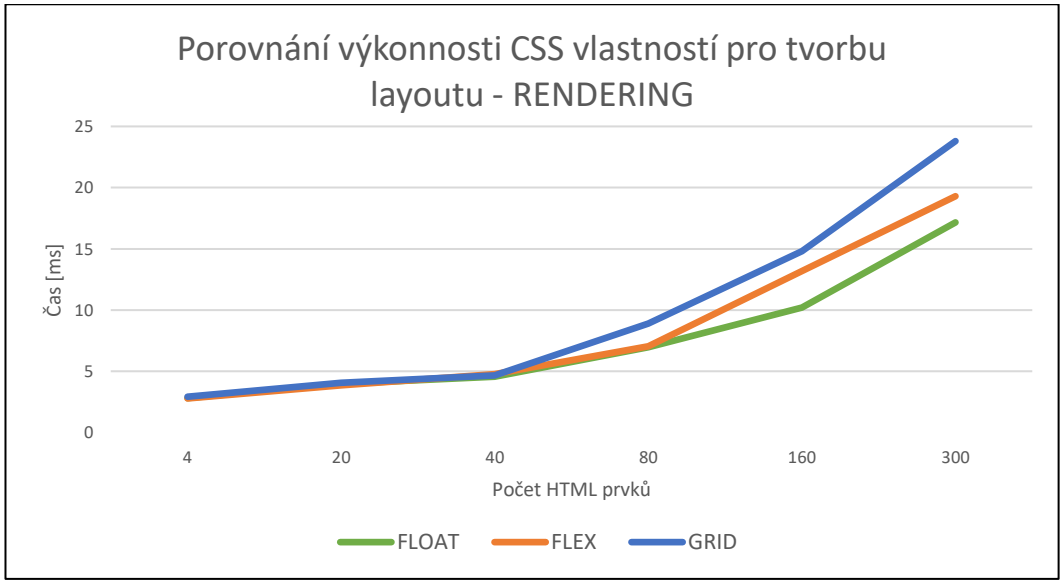
4.1.4 Zhodnocení naměřených výsledků layoutových technik.

Po provedení detailních měření se nám naskýtá ucelený obraz o rozdílném chování jednotlivých layoutových modelů. Celkově se podařilo zjistit, že vybrané layoutové techniky vykazují rozdílné doby načítání. Jako nejrychlejší layoutová technika se ukázal být float. Ten byl následován flexboxem. Jako nejpomalejší layoutová technika se ukázal být grid. Následující graf nám přehledně demonstruje komparativním způsobem, jak si jednotlivé vlastnosti, v závislosti na počtu testovaných prvků vedly.



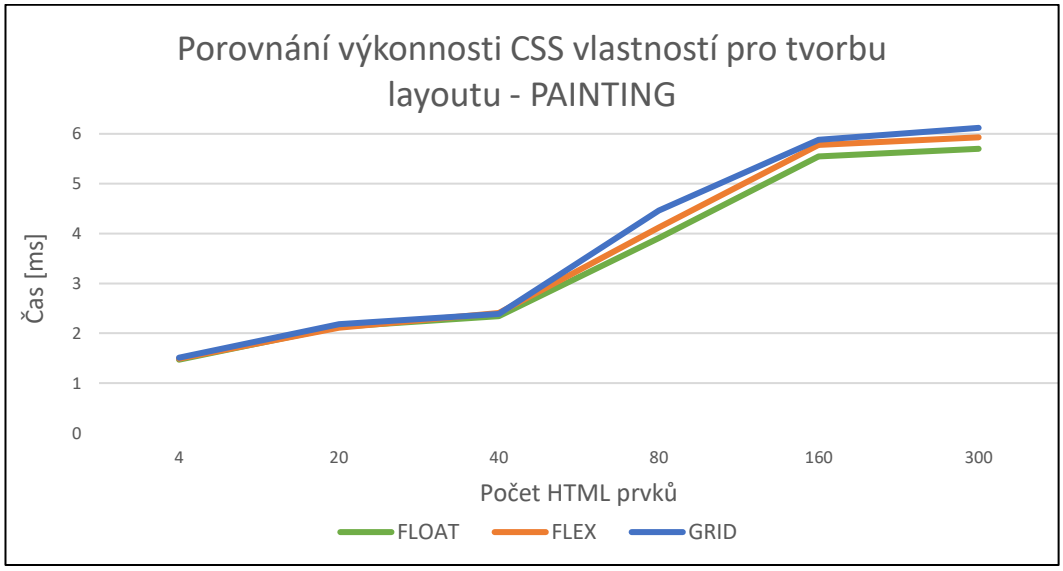
Graf č. 1: Porovnání výkonnosti layoutových vlastností (zdroj: autor práce)

U počtu prvků 4, 20 a 40 se vyskytují rozdílné, avšak stále velmi blízké hodnoty. Celkový čas vykreslení se začíná diferencovat od počtu prvků 80, kde lze vidět vlastnost grid, která jasně převyšuje ostatní vlastnosti. U počtu prvků 160 se od sebe začaly výrazně lišit všechny vybrané vlastnosti. V posledním zkoumaném množství 300 byl trend podobný jako v kroku předchozím s tím, že markantnější nárůst doby vykreslení zaznamenala vlastnost grid. Podrobnější zobrazení doby paintingu a renderingu je vidět na následujících dvou grafech. Ty se zvlášť zaměřují na dobu paintingu a renderingu pro jednotlivé počty prvků.



Graf č. 2: Porovnání výkonností layoutových vlastností – Rendering (zdroj: autor práce)

Graf č. 2 zobrazuje dobu renderingu pro všechny tři testované vlastnosti. Graf č. 3 zobrazuje dobu paintingu pro testované vlastnosti.



Graf č. 3: Porovnání výkonností layoutových vlastností – Painting (zdroj: autor práce)

Na první pohled je z obou grafů patrné, že vykazují odlišný trend. Zatímco u časů paintingu je možné vidět lineárně rostoucí trend, u renderingu je tomu přesně naopak, ten vykazuje trend exponenciální. Jinými slovy to naznačuje, že doba paintingu narůstá aditivně o stejné hodnoty čísel. U renderingu se dají tyto přírůstky vyjádřit násobky a proto je trend tak rostoucí. Zajímavá je také skutečnost, že do hodnoty 40 vykazovaly oba, jak rendering tak painting, poměrně konzistentní chování. Od této hodnoty pak nastalo zvýšení hodnot a také se od sebe jejich výkonosti začaly znatelně lišit.

4.2 Testování drahých vlastností CSS.

V moderním webdesignu se používají pokročilé CSS vlastnosti k vytvoření esteticky přitažlivých a dynamicky interagujících webových stránek. Tyto vlastnosti umožňují designérům a vývojářům vytvářet složitější vizuální efekty, ačkoli přidávají na vizuální atraktivitě, mohou s sebou nést významné ovlivnění rychlosti načítání stránek a celkového uživatelského prožitku. Některé CSS vlastnosti ovlivňují rychlost načítání stránky více než ostatní, tyto vlastnosti jsou nazývány jako takzvané „drahé“ vlastnosti CSS. Byl prozkoumán jejich dopad na rychlost načítání webových stránek a byl poskytnut ucelený přehled o tom, jak je efektivně a s minimálním dopadem na výkon implementovat. Konkrétně byly vybrány tyto

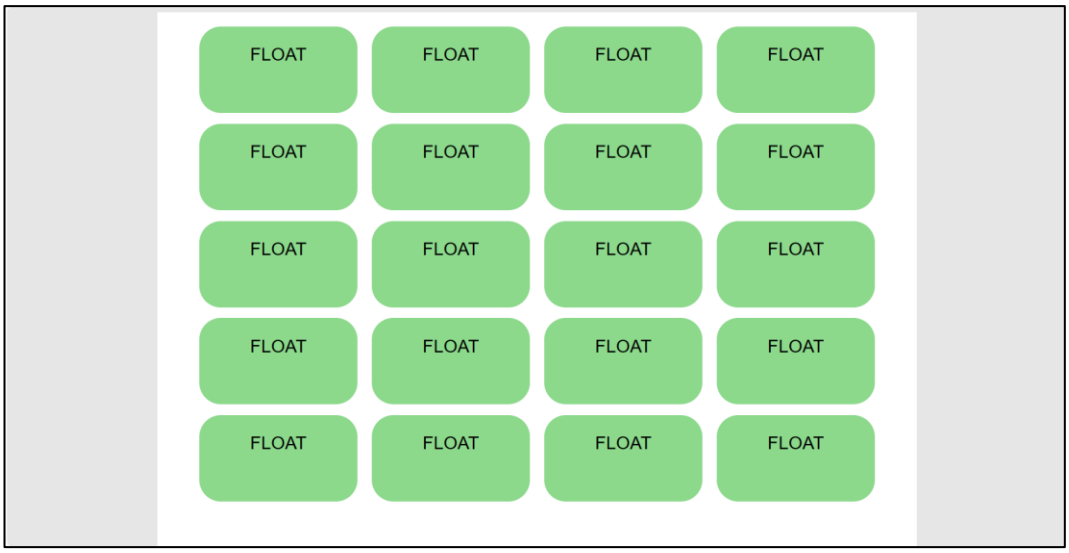
vlastnosti: border-radius, box-shadow, filter a transform. Pro měření dopadu těchto vlastností na rychlost načítání bylo vytvořeno několik testovacích webových stránek, na kterých byly tyto vlastnosti implementovány v různém rozsahu a intenzitě. Stejně jako při testování layoutových modelů byla zvolena jednotlivá měření obsahující následující počty: 4, 20, 40, 80, 160 a 300 prvků. Byly vytvořeny verze stránek s různými kombinacemi a úrovněmi komplexity použití border-radius, box-shadow, filter a transform, aby bylo možné analyzovat jejich individuální případně kombinovaný dopad. Výkonnost každé testovací stránky byla hodnocena pomocí nástrojů Chrome DevTools.

Při analýze drahých CSS vlastností byly vytvořeny celkem 4 testovací stránky, tedy pro každou testovanou vlastnost jedna stránka. Vždy se vycházelo z jednotné stuktury HTML dokumentu pro všechny testované stránky. Na každé testované stránce se opět navyšoval počet HTML prvků, pro které bylo měření prováděno. Pro každý počet bylo provedeno celkem 10 měření, ze kterých byl následně proveden aritmetický průměr. Aby mohla být provedena komparativní analýza vlivu jednotlivých drahých vlastností CSS, bylo zapotřebí si zvolit základní hodnoty, ze kterých se vycházelo. Jako základní hodnoty byly zvoleny HTML prvky s vlastností float z předchozí kapitoly testování layoutu. Ty sloužily jako základní hodnoty bez obsahu drahé vlastnosti. Následně byly přidávány jednotlivé drahé vlastnosti a byl zkoumán jejich vliv.

4.2.1 Vlastnost border-radius

Vlastnost border-radius umožňuje přidat kulaté rohy danému HTML prvku. Může nabývat čtyř hodnot, přičemž každá hodnota určuje jeden ze čtyř rohů zaoblení. Tyto hodnoty mohou být libovolné.[53] Pro měření vlastnosti border-radius byla nastavena hodnota na 30px. Na následujícím obrázku je vidět vizuální podoba jedné z testovacích stránek.

[53] NIEDERST ROBBINS, Jennifer. *Learning web design: a beginner's guide to HTML, CSS, Javascript, and web graphics* [online]. Fifth edition. Beijing: O'Reilly, 2018 [cit. 2024-01-25]. ISBN 978-1491960202. Dostupné z: https://learning.oreilly.com/library/view/learning-web-design/9781491960196/



Obr. č. 22: CSS kód vykreslen prohlížečem (zdroj: autor práce)

Všechny naměřené hodnoty, z každé ze šesti testovacích kol, je možné vidět v následující tabulce č. 4. V tabulce jsou vidět dvě hodnoty. Levá hodnota (bez použití vlastnosti) jak již bylo zmíněno, představuje základní hodnoty, vůči kterým jsou jednotlivá měřením posuzována. Pravé hodnoty představují naměřené hodnoty pro danou aktuální měřenou vlastnost. V posledním sloupci lze vidět procentuální rozdíl, tedy o kolik procent se celková doba vykreslení zhoršila nebo zlepšila.

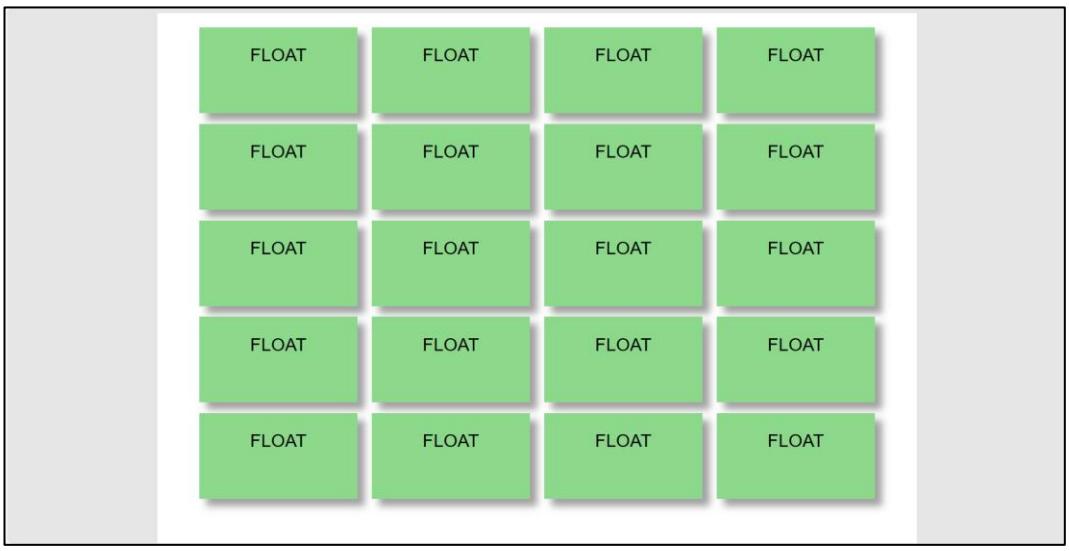
POČET HTML PRVKŮ	DOBA VYKRESLENÍ BEZ POUŽITÍ VLASTNOSTI [ms]	DOBA VYKRESLENÍ S POUŽITÍM VLASTNOSTI [ms]	ROZDÍL [ms]
4	4,29	4,31	0,5%
20	6,12	6,22	1,6%
40	6,90	7,04	2,0%
80	10,87	11,23	3,3%
160	15,75	16,18	2,7%
300	22,86	23,66	3,5%

Tab. č. 4: Výsledky testované vlastnosti border-radius (zdroj: autor práce)

4.2.2 Vlastnost box-shadow

Vlastnost box-shadow umožňuje přidat pro HTML element efekt stínu. Pro tuto vlastnost jsou základní dvě hodnoty, které udávají velikost odsazení stínu relativně k postavení HTML elementu. Dalšími volitelnými hodnotami jsou hodnoty blur, spread radius, a color.

Pomocí nich lze nastavovat další efekty stínu.[54] Pro toto měření byla vlastnost box-shadow nastavena na hodnoty 10px 10px 10px #808080bf. Stejně jako v předchozím případě bylo provedeno deset měření pro každý počet html prvků. Vizuální podobu použité vlastnosti je možné vidět na obrázku číslo 23. Naměřené hodnoty vlivu této vlastnosti lze pak vidět hned v tabulce následující za obrázkem.



Obr. č. 23: CSS kód vykreslen prohlížečem (zdroj: autor práce)

Na obrázku č. 23 je vidět testovaná vlastnost box-shadow pro 20 HTML prvků na stránce. V tabulce č. 5 jsou vidět naměřené hodnoty vykreslení bez a s použitou vlastností box-shadow.

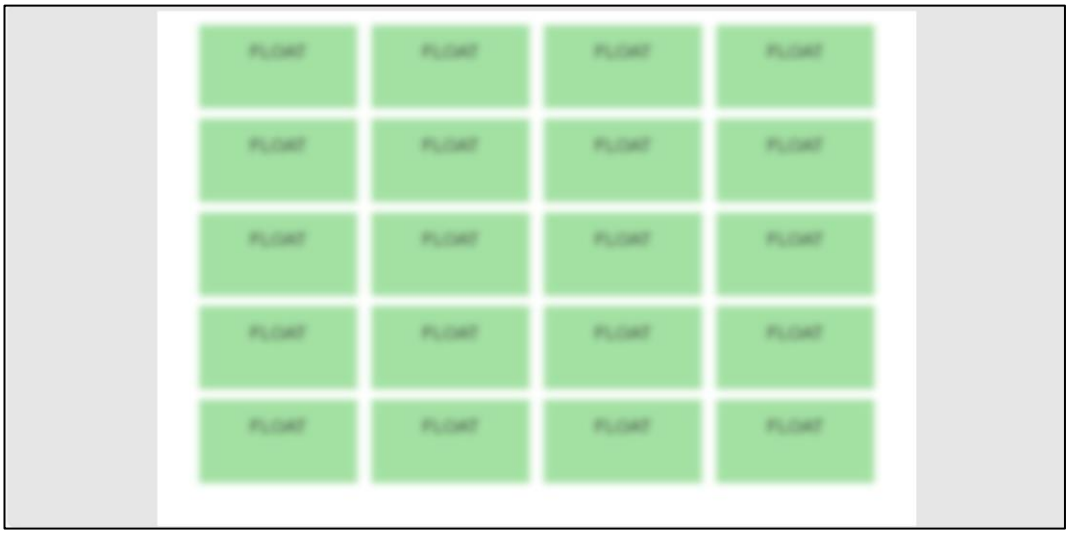
POČET HTML PRVKŮ	DOBA VYKRESLENÍ BEZ POUŽITÍ VLASTNOSTI [ms]	DOBA VYKRESLENÍ S POUŽITÍM VLASTNOSTI [ms]	ROZDÍL [ms]
4	4,29	4,38	2,1%
20	6,12	6,26	2,3%
40	6,90	7,10	2,9%
80	10,87	11,49	5,7%
160	15,75	16,87	7,1%
300	22,86	23,97	4,9%

Tab. č. 5: Výsledky testované vlastnosti box-shadow (zdroj: autor práce)

[54] MOZILLA, Developer. *Box-shadow*. Developer Mozilla [online]. 2024 [cit. 2024-01-15]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/CSS/box-shadow

4.2.3 Vlastnost filter

Vlastnost CSS filter na HTML prvek aplikuje grafické efekty. Tyto grafické efekty mohou být rozmazání, pootočení, změna kontrastu a další. Tato vlastnost má mnoho předdefinovaných funkcí, které je možné aplikovat. Z těch nejznámějších se jedná například o funkci blur(), opacity() nebo brightness().[55] Pro účel zkoumání vlivu této vlastnosti byla tato vlastnost definována se dvěma funkčními hodnotami, a to konkrétně opacity(0.8) a blur(5px). Vizuální podoba stránky a výsledné hodnoty měření jsou uvedeny na následujících obrázcích.



Obr. č. 24: CSS kód vykreslen prohlížečem (zdroj: autor práce)

Na obrázku č. 24 je vidět testovaná vlastnost filter pro 20 HTML prvků na stránce. V tabulce č. 6 jsou vidět naměřené hodnoty vykreslení bez a s použitou vlastností filter.

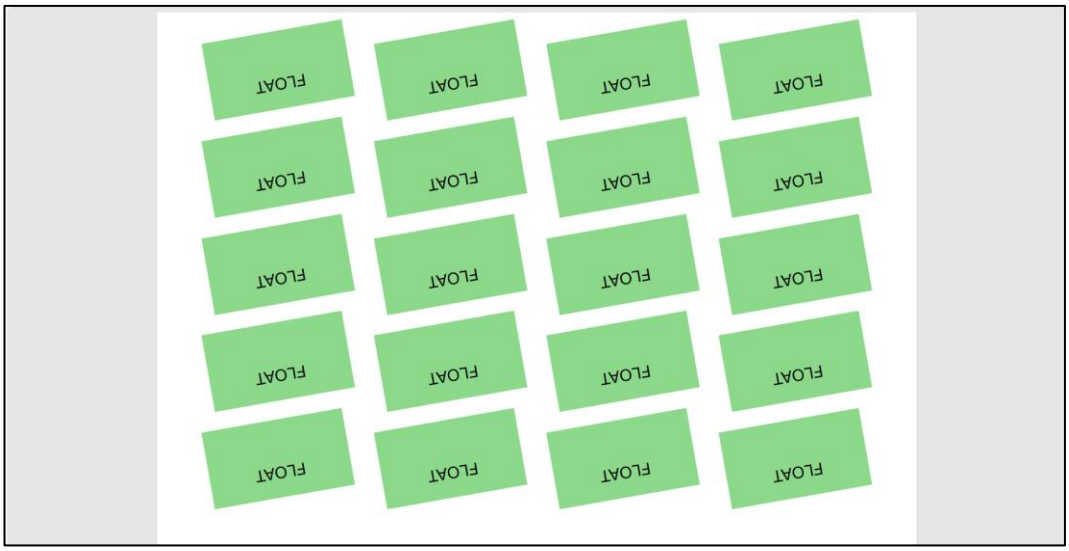
POČET HTML PRVKŮ	DOBA VYKRESLENÍ BEZ POUŽITÍ VLASTNOSTI [ms]	DOBA VYKRESLENÍ S POUŽITÍM VLASTNOSTI [ms]	ROZDÍL [ms]
4	4,29	4,57	6,5%
20	6,12	6,71	9,6%
40	6,90	7,56	9,6%
80	10,87	12,24	12,6%
160	15,75	17,31	9,9%
300	22,86	26,08	14,1%

Tab. č. 6: Výsledky testované vlastnosti filter (zdroj: autor práce)

[55] MOZILLA, Developer. *Filter*. Developer Mozilla [online]. 2024 [cit. 2024-01-15]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/CSS/filter.

4.2.4 Vlastnost transform

Jako poslední vlastnost byla pro testování výkonnosti vybrána CSS vlastnost transform. Vlastnost transform dovoluje otočit, zkosit nebo například změnit měřítko daného prvku. Má předdefinovanou sadu funkcí, které je možné aplikovat. Příkladem je například funkce rotate() nebo scale().[56] Pro účely měření byla vlastnost transform nastavena na hodnoty scale(0.9) a rotate(170deg). Vzor reálného zobrazení stránky je možné vidět na následujícím obrázku, stejně tak jako tabulku s naměřenými výslednými zprůměrovanými hodnotami.



Obr. č. 25: CSS kód vykreslen prohlížečem (zdroj: autor práce)

Na obrázku č. 25 je vidět testovaná vlastnost transform pro 20 HTML prvků na stránce. V tabulce č. 7 jsou vidět naměřené hodnoty vykreslení bez a s použitou vlastností transform.

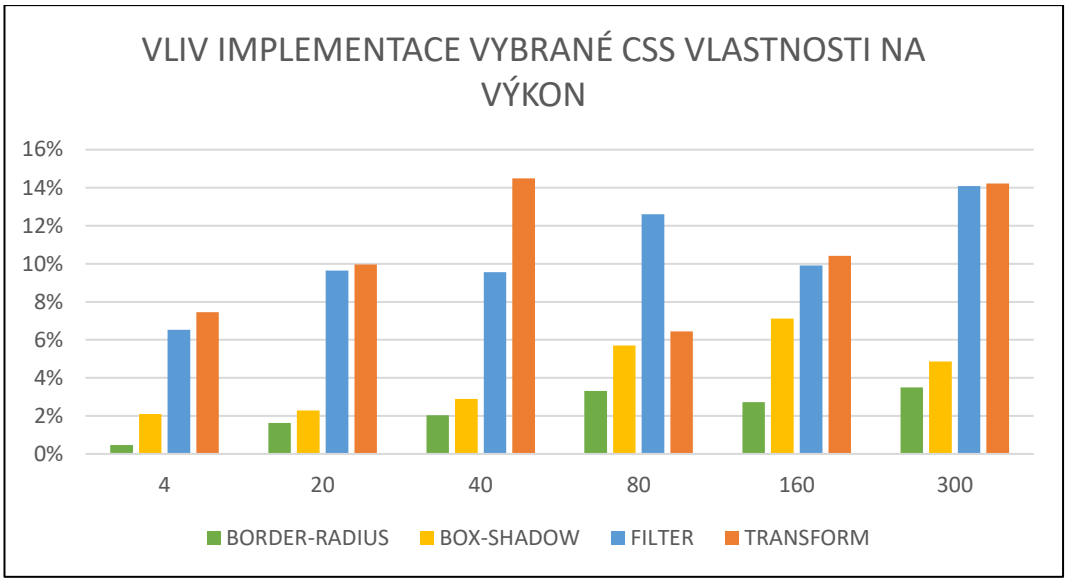
POČET HTML PRVKŮ	DOBA VYKRESLENÍ BEZ POUŽITÍ VLASTNOSTI [ms]	DOBA VYKRESLENÍ S POUŽITÍM VLASTNOSTI [ms]	ROZDÍL [ms]
4	4,29	4,61	7,5%
20	6,12	6,73	10,0%
40	6,90	7,90	14,5%
80	10,87	11,57	6,4%
160	15,75	17,39	10,4%
300	22,86	26,11	14,2%

Tab. č. 7: Výsledky testované vlastnosti transform (zdroj: autor práce)

[56] MOZILLA, Developer. *Transform*. Developer Mozilla [online]. 2024 [cit. 2024-01-15]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/CSS/transform.

4.2.5 Zhodnocení vlivu drahých vlastností:

Na základě provedené analýzy zaměřené na zkoumání vlivu drahých vlastností CSS na rychlost načítání webové stránky lze konstatovat, že každá z výše testovaných vlastností má svůj specifický dopad na výkonnost stránek. V průběhu testování se ukázalo, že použití border-radius, box-shadow, filter a transform má variabilní dopad na výkonnost stránek v závislosti na jejich intenzitě a způsobu implementace. To naznačuje, že při návrhu webů by mělo být bráno v potaz nejen vizuální působení, ale i výkonnostní aspekty. Při komparativní analýze jednotlivých vlastností je možné pozorovat, že všechny výše testované „drahé“ vlastností si vedly v době načítání hůře, než výchozí zvolená stránka. Jinými slovy lze říci, že všechny výše testované vlastnosti vykazovaly vyšší doby načítání a lze je považovat za náročnější. Mírný nárůst rychlost načítání můžeme pozorovat u vlastností border-radius a box-shadow. Vlastnost border-radius vykazuje v průměru o 2 % horší doby načítání než HTML prvky bez této vlastnosti. Vlastnost box-shadow vykazuje v průměru o 4 % horší dobu načítání než u HTML prvků bez testované vlastnosti. Vysoký nárůst lze naopak pozorovat u vlastností filter a transform. U těchto vlastností se dostáváme až ke dvojcifernému nárůstu doby načítání. U 4 a 20 testovaných prvků vykazovala mírně lepší dobu načítání vlastnost filter. Od počtu testovaných prvků 40 až po 300 nelze jednoznačně určit horší či lepší doby načítání. Vlastnosti byly velice proměnlivé a střídaly své první a druhé místo. V průměru vykazovaly obě vlastnosti horší dobu načítání o přibližně 10 % než definovaná základní stránka bez drahých vlastností CSS. To naznačuje, že i když tyto vlastnosti mohou přispět k vizuálně atraktivnímu designu, jejich nadměrné nebo neoptimalizované použití může negativně ovlivnit rychlost načítání webové stránky. Detailní výsledky jsou zobrazeny v následující tabulce. Na vertikální ose je vidět procentuální zastoupení, které nám říká o kolik je daná vlastnost horší než základní HTML prvky bez použití vlastností. Na horizontální ose je zobrazeno množství testovaných prvků.



Graf č. 4: Porovnání výkonu drahých vlastností CSS (zdroj: autor práce)

4.3 Dopad používání CSS frameworků na rychlost načítání webových stránek

V současném prostředí webového vývoje čelí vývojáři rostoucímu tlaku a očekáváním, která se týkají nejenom tvorby SEO optimalizovaných a responzivních webových stránek. Jsou kladeny čím dál větší požadavky na rychlost vývoje a spuštění těchto stránek. V tomto kontextu hrají CSS frameworky klíčovou roli. S jejich sadami předdefinovaných tříd a prvků, zjednodušují a zrychlují vytváření konzistentního, responzivního a flexibilního designu. Ačkoli tyto frameworky nabízí značné výhody, mohou s sebou nést i potenciální komplikace. Především pokud jde o efektivitu načítání webových stránek. Následující část práce je zaměřena na zkoumání vlivu aplikace frameworku pro vývoj webových stránek. Nabízí se otázka, zda je při tvorbě webu lepší použití samotných, „čistých“ CSS stylů, nebo použití CSS frameworků. Bude brán zřetel nejen na ovlivnění rychlosti načítání způsobené frameworkem, ale také na usnadnění tvorby a snížení doby psaní stylů, kterou použití frameworku přineslo.

V rámci této výzkumné části bylo přistoupeno k otázce vlivu CSS frameworků na rychlost načítání webových stránek systematicky. Nejprve byla pozornost věnována izolovanému experimentu, kde bylo sledováno, jak se mění rychlost načítání stránky s rostoucím počtem prvků. Tyto stránky byly vytvořeny a stylizovány pomocí CSS frameworků Bootstrap a Tailwind. Testování bylo započato s malým množstvím prvků, který se postupně zvyšoval až na 10 000, aby bylo patrné, jak se každý framework s nárůstem zatížení vyrovnává a který má

lepší výsledky. Tento přístup umožnil získat přesné údaje a hlubší pochopení dynamiky frameworků v kontextu jejich škálovatelnosti.

Byly vytvořeny dvě základní testovací webové stránky, na kterých se testovalo načítání jednoho prvku, jehož počet na stránce byl postupně navyšován od počtu 50, 100, 500, 1000, 5000 až na celkový počet 10 000. Pro každou verzi stránky byl použit buď framework Bootstrap nebo Tailwind. Veškerá měření proběhla opět pomocí nástroje Chrome DevTool. Byl kladen velký důraz na to, aby byl každý prvek stylován tak, aby vizuálně odpovídal oběma frameworkům a lze ho tak považovat za identický. Testování bylo provedeno za konzistentních podmínek, aby byly zajištěny relevantní a srovnatelné výsledky. Všechna měření byla pro jednotlivé množství prvků vždy provedena 10krát a poté zprůměrována. Výhodou používání frameworků je to, že není potřeba vést oddělené CSS soubor se styly a všechny styly je možné implementovat rovnou pomocí předdefinovaných názvů tříd do jednoho HTML souboru.

4.3.1 Tailwind

Pro testování frameworku Tailwind byla vytvořena základní struktura HTML stránky, která v sobě obsahovala kaskádové styly v podobě předdefinovaných tříd frameworkem Tailwind.

```
15      <div class="container mx-auto px-4">
16        <div class="py-4 mt-2 bg-blue-500 text-white rounded-lg px-5">
17          Prvek 1
18        </div>
19        <div class="py-4 mt-2 bg-blue-500 text-white rounded-lg px-5">
20          Prvek 2
21        </div>
22        <div class="py-4 mt-2 bg-blue-500 text-white rounded-lg px-5">
23          Prvek 3
24        </div>
25        <div class="py-4 mt-2 bg-blue-500 text-white rounded-lg px-5">
26          Prvek 4
27        </div>
28        <div class="py-4 mt-2 bg-blue-500 text-white rounded-lg px-5">
29          Prvek 5
30        </div>
```

Obr. č. 26: HTML kód s Tailwind CSS pro testování frameworku Tailwind (zdroj: autor práce)

Na obrázku č. 26 můžeme vidět strukturu kódu stránky zhotovené za pomocí frameworku Tailwind. Vizuální podobu stránky je vidět na obrázku č. 27.

Obr. č. 27: Zobrazení testovací stránky pro framework Tailwind prohlížečem (zdroj: autor práce)

Všechny výsledné naměřené hodnoty byly pro jednotlivé množství prvků zprůměrovány a rozděleny podle času renderingu a paintingu. V poslední části je možné vidět celkový čas vykreslení. Tabulka je uvedena níže.

POČET HTML PRVKŮ	RENDERING [ms]	PAINTING [ms]	CELKEM [ms]
50	7,32	2,45	9,77
100	13,39	3,27	16,66
500	21,73	4,56	26,29
1000	58,51	9,14	67,65
5000	215,66	14,43	230,09
10000	381,77	17,86	399,63

Tab. č. 8: Výsledky při využití frameworku Tailwind (zdroj: autor práce)

4.3.2 Bootstrap

Identickým způsobem byla vytvořena druhá testovací stránka pro framework bootstrap. Výsledná měření byla provedena a zaznamenána stejným způsobem jako u frameworku Tailwind.

```
<div class="container-fleak ml-3 mr-3">
  <div class="p-3 mt-2 bg-primary text-white mt-1 rounded">
    Prvek 1
  </div>
  <div class="p-3 mt-2 bg-primary text-white mt-1 rounded">
    Prvek 2
  </div>
  <div class="p-3 mt-2 bg-primary text-white mt-1 rounded">
    Prvek 3
  </div>
  <div class="p-3 mt-2 bg-primary text-white mt-1 rounded">
    Prvek 4
  </div>
  <div class="p-3 mt-2 bg-primary text-white mt-1 rounded">
    Prvek 5
  </div>
```

Obr. č. 28: HTML kód s Bootstrap CSS pro testování frameworku Bootstrap (zdroj: autor práce)

Na obrázku č. 28 je zobrazen HTML kód stránky. Vizuální podobu testované stránky je možné vidět na obrázku č. 29.

Obr. č. 29: Zobrazení testovací stránky pro framework Bootstrap prohlížečem (zdroj: autor práce)

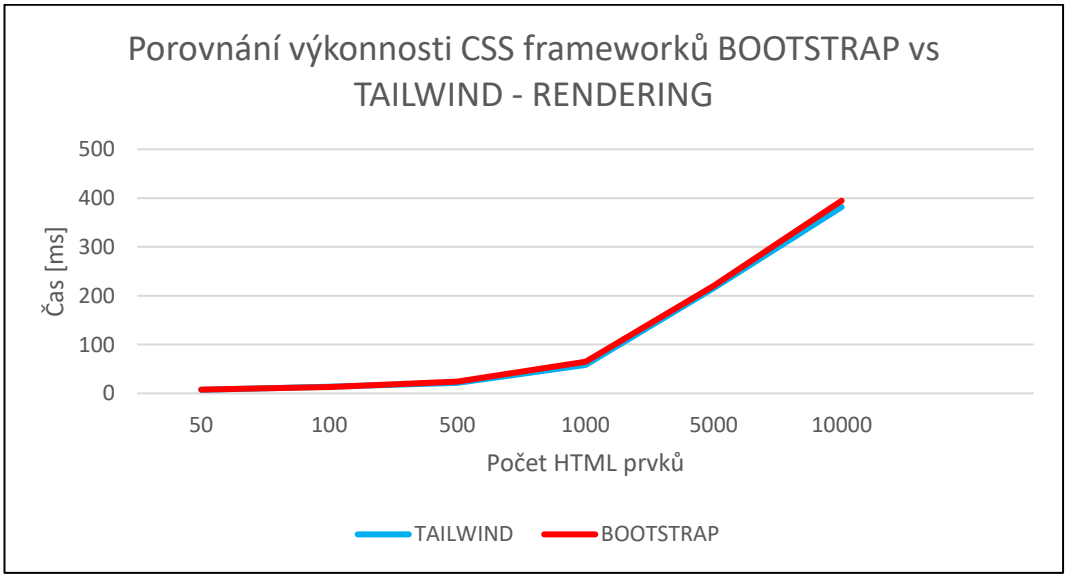
Výsledné naměřené hodnoty renderingu a paintingu pro testovanou stránku je možné vidět na následující tabulce č. 9.

POČET HTML PRVKŮ	RENDERING [ms]	PAINTING [ms]	CELKEM [ms]
50	7,81	2,51	10,32
100	13,31	3,96	17,27
500	24,49	4,86	29,35
1000	65,11	9,57	74,68
5000	220,07	14,84	234,91
10000	394,72	18,99	413,71

Tab. č. 9: Výsledky při využití frameworku Bootstrap (zdroj: autor práce)

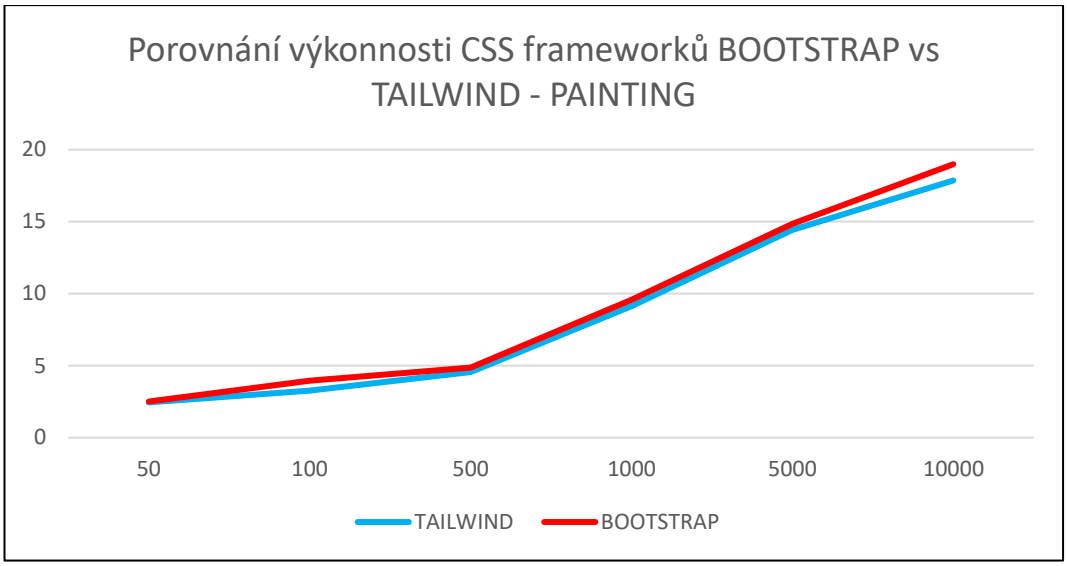
4.3.3 Zhodnocení využití frameworku Bootstrap a Tailwind

Analýza byla založena na měření rychlosti načítání různého počtu HTML prvků vytvořených za pomocí dvou CSS frameworků, od 50 až do 10 000, což poskytlo důkladný pohled na škálovatelnost obou frameworků. Následující grafy detailně ilustrují jak si oba frameworky vedly.



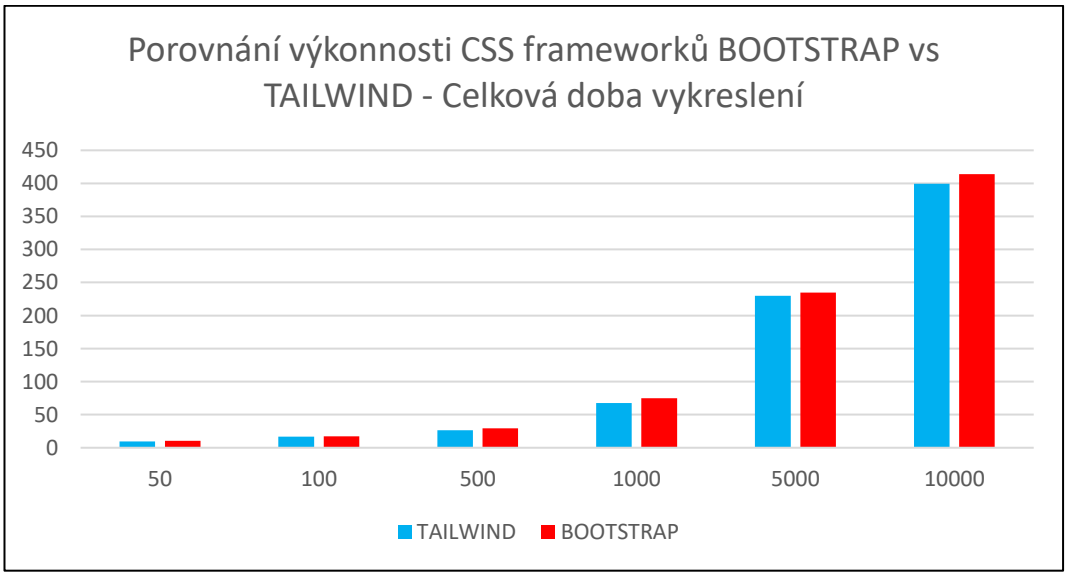
Graf č. 5: Výkonnost frameworků – Rendering (zdroj: autor práce)

V prvním grafu (č. 5) je možné vidět porovnání obou frameworků v rámci času renderingu. V dalším grafu (č. 6) je zobrazena doba paintingu.



Graf č. 6: Výkonnost frameworků – Painting (zdroj: autor práce)

Na Grafu č. 7 je zobrazena celková výsledná doba vykreslení pro oba testované frameworky.



Graf č. 7: Porovnání výkonnosti obou frameworků (zdroj: autor práce)

Z naměřených dat je zřejmé, že Tailwind konzistentně překonává Bootstrap ve všech testovaných kategoriích, a to jak v renderingu, tak v paintingu. Zvláště pak při vyšším počtu prvků se tyto rozdíly stávají signifikantní. Například při testování s 10 000 prvky byla celková doba načítání pro Tailwind nižší o 13,08 ms ve srovnání s Bootstrapem, což naznačuje lepší optimalizaci a výkon při práci s velkými objemy obsahu. Důležitým zjištěním je, že s rostoucím počtem prvků se zvyšuje i celkový čas potřebný pro načtení stránky, avšak růst je u Tailwindu

mírnější. To může být přisouzeno jeho designové filozofii zaměřené na utility-first přístup, který přináší méně redundantního kódu a efektivnější CSS selektory. Tailwind také umožňuje vývojářům používat jen ty styly, které jsou skutečně potřebné pro dané komponenty, což může výrazně snížit velikost konečného CSS souboru. Naopak Bootstrap, ačkoli poskytuje robustní seznam předdefinovaných komponent, může v důsledku toho vést k většímu množství nevyužitého CSS, což má za následek delší doby načítání. Tento rozdíl může mít významný dopad na uživatelskou zkušenost, zejména na zařízeních s omezeným výkonem nebo při pomalém internetovém připojení.

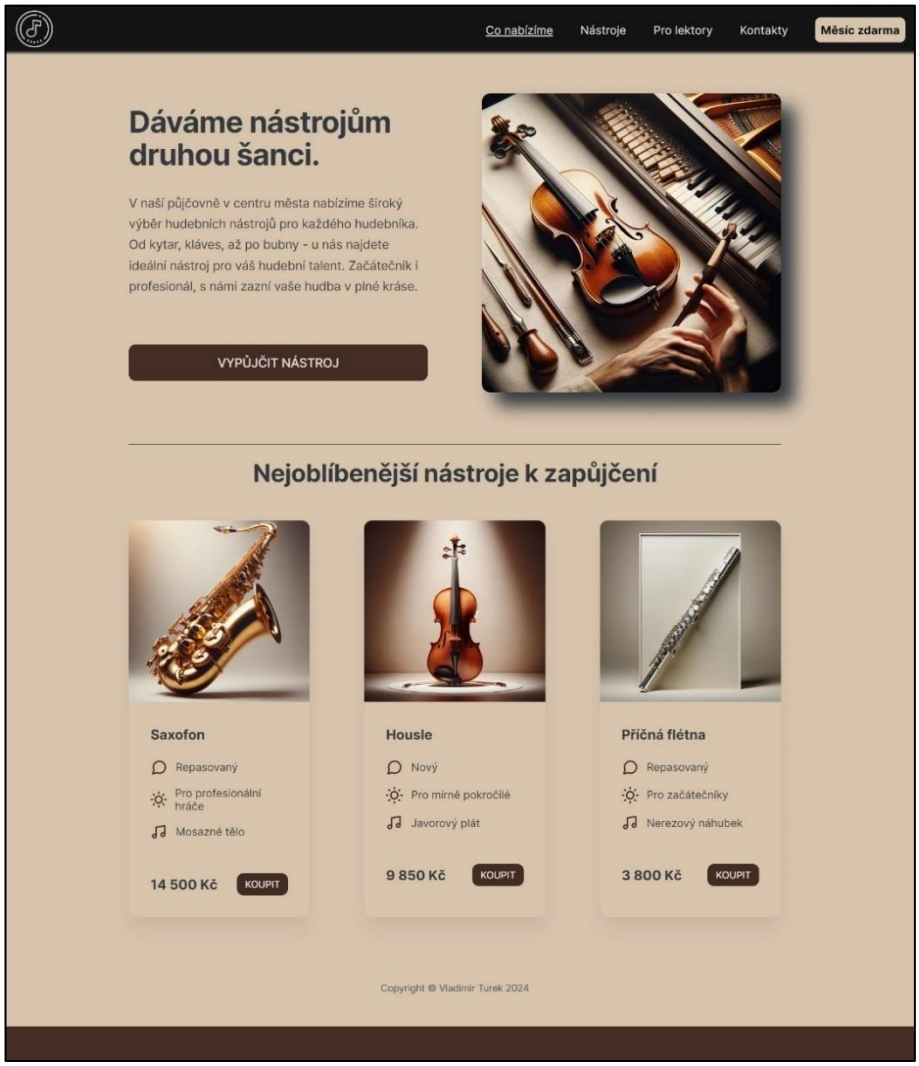
4.4 Analýza výkonu reprezentativní webové stránky

4.4.1 „Čisté“ styly v porovnání s CSS frameworkem Tailwind

Tato kapitola byla věnována komplexní analýze výkonu reprezentativní webové stránky. Bylo přistoupeno k souhrnnému testování, kdy framework Booststrap, který se ukázal být dle provedených měření v předchozí kapitole rychlejší, byl aplikován na konkrétní vytvořenou webovou stránku. Tento krok byl v posloupnosti celé praktické části zásadní, neboť umožnil posoudit celkový dopad na načítání stránky v kontextu reálného scénáře. Nejednalo se pouze o zpracování velkého množství prvků, ale o konkrétní zjištění, jak frameworky ovlivní načítání různorodého obsahu od textu a obrázků až po složitější komponenty a také jak usnadňují proces tvorby webových stránek. Vytvořená stránka byla navržena s cílem prezentovat služby firmy nabízející půjčení hudebních nástrojů veřejnosti, studentům uměleckých škol a lektorům. Byla vyvinuta ve dvou verzích. V prvním případě použitím čistého HTML a CSS, tedy bez použití frameworku. Ve druhém případě s využitím CSS frameworku Tailwind, což umožnilo přímé srovnání vlivu na výkon webu. Výběr frameworku Tailwindu byl motivován předchozími zjištěními, která ukázala jeho rychlost ve srovnání s jinými frameworky. V rámci tohoto výzkumu byla pozornost zaměřena na metriky Core Web Vitals což umožnilo kvantitativně hodnotit a porovnat výkonnost obou variant webové stránky.

Pro analýzu výkonu byly vybrány klíčové webové metriky. Těmito metrikami byly Largest Contentful Paint (LCP), First Input Delay (FID), Cumulative Layout Shift (CLS), First Contentful Paint (FCP), Interaction to Next Paint (INP) a Time to First Byte (TTFB). Tyto metriky byly využity s cílem poskytnout ucelený pohled na rychlost načítání, vizuální stabilitu a na rychlost požadavků putujících od webového serveru. K získání těchto dat byl použit nástroj

PageSpeed Insights od Google, což umožnilo přesné a konzistentní měření výkonu v reálných podmínkách použití. Měření byla opět provedena v konzistentním prostředí, aby byla zajištěna přesnost a srovnatelnost výsledků. Reprezentativní webová stránka je viditelná na obrázku č. 30.



Obr. č. 30: Reprezentativní webová stránka (zdroj: autor práce)

Testovací měření bylo provedeno stejně jako v předchozích kapitolách v deseti opakováních pro každou metriku měření, jak pro stránku s čistými styly, tak pro stránku za použití frameworku. Výsledné hodnoty byly zprůměrovány. Demonstrující naměřené hodnoty jsou uvedeny v následujících dvou tabulkách. V tabulce č. 10 jsou bazické hodnoty, které slouží jako základní porovnávací hodnoty pro srovnání využití frameworku. Tabulka obsahuje naměřené hodnoty s využitím „čistého“ CSS bez využití frameworků.

ZVOLENÁ METRIKA	NAMĚŘENÝ ČAS [s]
Largest Contentful Paint (**LCP**)	1,400
First Input Delay (**FID**)	0,012
Cumulative Layout Shift (**CLS**)	0,000
First Contentful Paint (**FCP**)	1,300
Interaction to Next Paint (**INP**)	0,176
Time to First Byte (**TTFB**)	1,000

Tab. č. 10: Naměřené hodnoty za použití čistého CSS (zdroj: autor práce)

Tabulka č. 11 obsahuje naměřené hodnoty pro testovanou webovou stránku za použití frameworku Tailwind.

ZVOLENÁ METRIKA	NAMĚŘENÝ ČAS [s]
Largest Contentful Paint (**LCP**)	1,500
First Input Delay (**FID**)	0,028
Cumulative Layout Shift (**CLS**)	0,000
First Contentful Paint (**FCP**)	1,470
Interaction to Next Paint (**INP**)	0,104
Time to First Byte (**TTFB**)	1,000

Tab. č. 11: Naměřené hodnoty za použití frameworku Tailwind (zdroj: autor práce)

Ve výsledné tabulce č. 12. jsou porovnané oba výše naměřené způsoby aplikace stylů. Poslední sloupeček tabulky zobrazuje rozdíly v měření.

ZVOLENÁ METRIKA	NAMĚŘENÝ ČAS BEZ POUŽITÍ FRAMEWOROKU [s]	NAMĚŘENÝ ČAS S POUŽITÍM FRAMEWORKU [s]	ROZDÍL [ms]
Largest Contentful Paint (**LCP**)	1,400	1,500	0,100
First Input Delay (**FID**)	0,012	0,028	0,016
Cumulative Layout Shift (**CLS**)	0,000	0,000	0,000
First Contentful Paint (**FCP**)	1,300	1,470	0,170
Interaction to Next Paint (**INP**)	0,176	0,104	-0,072
Time to First Byte (**TTFB**)	1,000	1,000	0,000

Tab. č. 12: Porovnání obou způsobů využití CSS (zdroj: autor práce)

Při porovnání naměřených hodnot obou verzí reprezentativní webové stránky bylo zjištěno, že stránka vyvinutá s použitím čistého CSS dosahuje lepších výsledků u většiny sledovaných metrik. Konkrétně metriky LCP a FCP byly nižší pro verzi bez frameworku, což

pro uživatele naznačuje rychlejší vizuální dostupnost obsahu. Také metrika FID, která měří reaktivitu stránky na první interakci uživatele, byla lepší (nižší) u verze bez použití Tailwindu, což poukazuje na vyšší uživatelský prožitek při interakci se stránkou.

Překvapivě, navzdory předchozím zjištěním o výhodách rychlosti Tailwindu, data ukazují, že pro komplexní webovou prezentaci může být přístup s čistým CSS výhodnější. Tento rozdíl může být částečně vysvětlen přidanou komplexitou, kterou s sebou Tailwind ve formě utility tříd přináší. Ty mohou zvýšit velikost finálního CSS a tím zpomalit načítání stránky.

Na základě provedené analýzy lze konstatovat, že výběr technologie pro vývoj webové stránky má zásadní vliv na její výkon. I když Tailwind nabízí výhody v rychlosti vývoje a flexibilitě, může být pro projekty, kde je prioritou maximální výkon, efektivnější použití čistého CSS. Měření naznačuje, že při pečlivé implementaci a optimalizaci může být čisté CSS schopno dosáhnout rychlejšího načítání stránky, nižšího First Input Delay a lepšího celkového uživatelského prožitku. Tento výsledek je zvláště důležitý v kontextu webových standardů a SEO.

Je důležité podotknout, že výběr mezi použitím frameworku jako Tailwind a psaním čistého CSS by neměl být považován za dogma. Místo toho by měl být webový vývoj přizpůsoben konkrétním potřebám projektu, cílové skupině a prostředí, ve kterém bude web používán. Pro rychle se vyvíjející projekty s krátkými časovými rámci může být Tailwind CSS vhodnější díky své modularitě a snadné přizpůsobitelnosti.

4.4.2 Vliv připojení CSS souboru na rychlost načítání

Volba způsobu, jakým CSS styly do webové stránky implementujeme (připojíme) a druh metody psaní stylů jsou důležitým rozhodnutím, jenž má dopady nejen na výkonnost a načítání stránky, ale také na její údržbu, škálovatelnost a celkový uživatelský prožitek. Jak již bylo zmíněno v teoretické části bakalářské práce, CSS můžeme do HTML stránky importovat hned několika způsoby. Buď externě, tedy připojením externího souboru, přiřazením stylu konkrétnímu prvku, který chceme stylovat, nebo v hlavičce. Avšak samotný způsob připojení souboru není jediný faktor mající vliv na rychlost načítání, dalšími může být samotná struktura daného souboru nebo jeho rozdělení do více menších souborů. Rozhodnutí mezi použitím jednoho velkého CSS souboru pro všechny stránky nebo více menších souborů, stejně i jako

volba komponentového psaní CSS nebo kritické metodiky psaní CSS může mít na rychlost načítání vliv a proto i tato skutečnost vyžaduje pečlivé zvážení.

Použití jednoho velkého CSS souboru může na první pohled působit jako nejjednodušší varianta, jelikož minimalizuje počet HTTP požadavků potřebných k načtení stylů stránky. Avšak tato metoda může vést ke zbytečnému načítání nepoužívaných stylů, což zatěžuje prohlížeč a zpomaluje dobu načítání. Na druhou stranu, rozdělení CSS do více souborů, které jsou načítány pouze tam, kde jsou potřeba, může načítání stránek zefektivnit tím, že se minimalizuje množství přenesených dat. Tento přístup však vyžaduje pokročilejší správu závislostí stylů a může zkomplikovat jejich údržbu.

Tato kapitola se věnuje podrobné analýze tří různých přístupů k implementaci CSS. Použití jednotlivých souborů CSS pro každou stránku na webu, použití jednoho velkého CSS souboru pro celý web a rozdělení stylů do více souborů CSS pro jednu stránku webu. Cílem je zjistit, který z těchto přístupů nabízí nejlepší výkon z hlediska rychlosti načítání. Pro účely srovnání byly vytvořeny tři verze testovacího webu. Stále se jedná o ten samý web, který byl již prezentován v předchozí kapitole. V této kapitole byla měřena pouze úvodní hlavní stránka, která byla ukázána v předešlé kapitole. Každá verze reprezentuje jeden z výše zmíněných přístupů k organizaci CSS. Aby bylo možné získat přesné a objektivní údaje o výkonu každé z těchto metod, byl využit nástroj PageSpeed Insights od Google. Tento nástroj umožnil měřit a analyzovat klíčové výkonnostní metriky. Díky PageSpeed Insights bylo možné získat důkladný přehled o tom, jak organizace CSS ovlivňuje načítání stránek a bylo získáno kvantifikovatelné srovnání mezi jednotlivými přístupy. Měření byla provedena za konzistentních podmínek, aby byla zajištěna maximální srovnatelnost výsledků mezi všemi testovanými verzemi webu. Na obrázku č. 31 je možné vidět uspořádání do jednoho CSS souboru pro každou webovou stránku. Na obrázku č. 32 je možné vidět uspořádání stylů tím způsobem, že jsou rozděleny dle komponent úvodní webové stránky.

```
<link rel="stylesheet" href="homepage.css" />
```

Obr. č. 31: Implementace CSS pomocí jednoho souboru pro stránku (zdroj: autor práce)

```
14    <link rel="stylesheet" href="header.css" />
15    <link rel="stylesheet" href="section.css" />
16    <link rel="stylesheet" href="footer.css" />
17    <link rel="stylesheet" href="nav.css" />
```

Obr. č. 32: Implementace CSS pomocí komponentového přístupu (zdroj: autor práce)

Opět byla veškerá měření 10krát opakována, poté zprůměrována a zaznamenána do tabulky. Výsledky je možné vidět na následující tabulce:

ZVOLENÁ METRIKA	NAMĚŘENÝ ČAS S JEDNÍM CSS SOUBOREM [s]	NAMĚŘENÝ ČAS S VÍCE CSS SOUBORY [s]	ROZDÍL [ms]
Largest Contentful Paint (**LCP**)	1,400	1,600	0,200
First Input Delay (**FID**)	0,012	0,031	0,019
Cumulative Layout Shift (**CLS**)	0,000	0,000	0,000
First Contentful Paint (**FCP**)	1,300	1,480	0,180
Interaction to Next Paint (**INP**)	0,176	0,137	-0,039
Time to First Byte (**TTFB**)	1,000	1,000	0,000

Tab. č. 13: Připojení více souborů CSS (zdroj: autor práce)

Pro účely druhé části měření byly všechny styly za celý web (pro všechny stránky) zahrnuty do jednoho souboru a používány přes celý web. Nově byl tedy připsán jeden velký CSS soubor obsahující veškeré CSS vlastnosti za celý web. Ukázka a naměřené hodnoty jsou na následujícím obrázku č. 33 a tabulce č. 14.

```
<link rel="stylesheet" href="vsevjednom.css" />
```

Obr. č. 33: Implementace CSS pomocí jednoho souboru pro celý web (zdroj: autor práce)

Opět byla veškerá měření 10krát opakována, poté zprůměrována a zaznamenána do tabulky. Výsledky je možné vidět na následující tabulce (č. 14).

ZVOLENÁ METRIKA	NAMĚŘENÝ ČAS S JEDNÍM CSS SOUBOREM [s]	NAMĚŘENÝ ČAS SLOUČENÉ STYLY [s]	ROZDÍL [ms]
Largest Contentful Paint (**LCP**)	1,400	1,500	0,100
First Input Delay (**FID**)	0,012	0,019	0,007
Cumulative Layout Shift (**CLS**)	0,000	0,000	0,000
First Contentful Paint (**FCP**)	1,300	1,420	0,120
Interaction to Next Paint (**INP**)	0,176	0,170	-0,006
Time to First Byte (**TTFB**)	1,000	1,000	0,000

Tab. č. 14: Styly sloučené do jednoho souboru (zdroj: autor práce)

4.4.3 Zhodnocení vlivu CSS souborů

Analýza odhalila, že nejrychlejšího načítání stránek bylo dosaženo, pokud každá stránka webu používala svůj vlastní dedikovaný CSS soubor. Tento přístup minimalizoval množství nevyužitého CSS, které prohlížeč musel zpracovat při načítání konkrétní stránky. To vedlo k rychlejšímu renderování obsahu. Na druhé příčce se umístilo použití jednoho velkého CSS souboru pro celý web. Ačkoli tento přístup zvyšuje velikost přenášených dat při prvním načtení, prohlížeče mohou tento soubor ukládat do mezipaměti a využívat jej pro rychlejší načítání dalších stránek webu. Jako nejpomalejší se ukázalo být rozdělení stylů do více souborů CSS pro jednu stránku, což způsobilo značné zpomalení v důsledku mnohonásobných požadavků na server a zvýšení času potřebného pro analýzu a aplikaci stylů.

5 Výsledky a diskuse

Tato kapitola je zaměřena na detailní zhodnocení a diskusi výsledků získaných z provedeného výzkumu, který byl zaměřen na rozličné vlastnosti a nástroje CSS, aspekty webového designu, výkon a optimalizaci webových stránek. Po prozkoumání různých layoutových technik pomocí experimentálních měření, po následném výběru a analýze konkrétních vlastností CSS, po zjištění role frameworku při tvorbě webových stránek, změření vlivu připojení CSS souborů a analýze reprezentativní webové stránky pomocí Web Vitals bylo odhaleno několik klíčových poznatků.

5.1 Výsledky použitých layoutových technik CSS

V rámci zkoumání vlivu layoutových technik CSS na rychlost načítání webové stránky, konkrétně vlastnosti float, flexbox a grid bylo zjištěno, že každá z těchto vlastností působí na načítání stránek rozdílně. Jako nejrychlejší techniky se pro malé množství testovaných prvků (konkrétně 4 a 20) na stránce ukázaly být layoutové techniky float a flexbox. Avšak do množství prvků 40, od sebe jednotlivé vlastnosti nevykazovaly významné rozdíly. Při zahrnutí vyššího množství prvků (80 až 300) je už rozdíl mezi jednotlivými vlastnostmi znatelný. Jako nejpomalejší se ukázal být ve všech případech grid. Float a flexbox při počtu prvků 80 vykazují prakticky stejný výkon. Od počtu 160 už si prvenství obhájil float, následovaný flexboxem.

Na základě poznatků je možné přisoudit prvenství v rychlosti layoutové technice float, která se ukázala být nejrychlejší. Pro webové stránky vyžadující komplikovanější layout s vyšší mírou personalizace do počtu prvků 40 se nabízí jako vhodná varianta i vlastnost flexbox.

I přesto, že layoutová vlastnost float může nabízet určité výhody v rychlosti načítání, její volba je vhodná především pro jednodušší webové stránky, případně ostatní konkrétní případy, kde není možná volba jiné vlastnosti. Integrace této vlastnosti přichází s omezeními v oblasti responzibility a flexibility. To činí jeho použití pro moderní web poměrně neefektivní, kdy se nepatrná úspora v rychlost načítání projeví mnohonásobným navýšením času a úsilí pro tvorbu webu, z hlediska jeho neintuitivního použití. Z tohoto důvodu se jeví jako vhodné při komplikovanějším layoutu také použití vlastnosti felxbox a jen v krajním případě použít techniku grid. Naměřené výsledky mohou být zapříčiněny simplifikovanou strukturou webové stránky zvolené pro testování.

5.2 Výsledky implementace drahých vlastností CSS

Vlastnosti border-radius, box-shadow, filter a transform byly vybrány z důvodu jejich velké obliby ve webovém designu a díky jejich potenciálnímu negativnímu dopadu na rychlost načítání webových stránek. U všech testovaných vlastností lze říci, že ovlivnily rychlost načítání, jinými slovy řečeno, při jejich implementaci došlo ke zhoršení času načítání.

Vlastnost s nejnižším dopadem na výkon je dle provedených testování vlastnost border-radius. Tedy k vykreslení zaoblených rohů touto vlastností není třeba velký výpočetní výkon. Jako druhá nejrychlejší byla identifikovaná vlastnost pro tvorbu stínovaní box-shadow. Což naznačuje že, i přeš její schopnost přidávat atraktivní vizuální vzhled jednotlivým prvkům na stránce se není třeba obávat negativních vlivů. Složitější vizuální transformace pomocí vlastností filter a transform vykázaly již větší dopad na dobu načítání než výše zmíněné vlastnosti border-radius a box-shadow. U obou testovaných vlastností bylo možné pozorovat vysoký nárůst doby načítání.

Dosažené výsledky poukazují na důležitost pečlivého zvažování použití „drahých“ CSS vlastností při tvorbě webového designu. Zatímco vlastnosti border-radius a box-shadow s minimálním dopadem na výkon poskytují atraktivní vizuální efekty, složitější efekty pomocí filter a transform již vyžadují rozvahu. Při používání těchto efektů je nutné najít rovnováhu.

5.3 Tailwind vs Bootstrap

Výzkumná část zaměřená na porovnání vlivu frameworků Tailwind a Bootstrap na rychlost načítání webové stránky přinesla řadu klíčových poznatků. Framework Tailwind se se svým utility-first přístupem ukázal být vhodnější. Tyto výsledky se dají vysvětlit tím, že Tailwind dává vývojářům lepší kontrolu nad stylováním a dovolí vývojářům přizpůsobit design zcela dle vlastních požadavků. Můžeme tak aplikovat pouze ty styly, které jsou skutečně potřebné, a to pomůže redukovat velikost CSS souboru.

Přestože Tailwind dle provedených měření nabízí výhody v rychlosti načítání, je důležité si uvědomit, že výběr frameworku nezávisí pouze na výkonu. Bootstrap nabízí cenné výhody pro projekty, které vyžadují urychlený proces vývoje a mají potřebu využít širokou škálu předdefinovaných komponent bez nutnosti hlubšího zasahování do stylování.

5.4 Reprezentativní webová stránka

Tato sekce se soustředila na podrobnou analýzu výkonnosti reprezentativní vytvořené webové stránky. Tato stránka byla analyzována pomocí klíčových metrik Web Vitals: Largest Contentful Paint (LCP), First Input Delay (FID), Cumulative Layout Shift (CLS), First Contentful Paint (FCP), Interaction to Next Paint (INP) a Time to First Byte (TTFB). Cílem bylo posoudit rozdíly v efektivitě mezi použitím čistého CSS a frameworku Tailwind CSS.

Na základě analýzy bylo odhaleno, že čisté CSS přineslo lepší výsledky ve většině sledovaných metrik, což naznačuje větší efektivitu a optimalizaci ve zpracování a vykreslení obsahu stránky. Zvláště významný byl rozdíl v metrikách LCP a FCP, což značí rychlejší načítání obsahu a vyšší plynulost, tedy tím pádem lepší uživatelský prožitek při prvním načtení stránky.

Čisté CSS se ukázalo být rychlejší v kontextu naší zjednodušené webové stránky. Pro projekty s menším množstvím stylů nebo specifickými požadavky na výkon má čisté CSS potenciál být výkonnější. Jedním z důvodů je, že přímé psaní CSS umožňuje větší kontrolu nad kódem a minimalizuje nadbytečné styly. Na druhou stranu, čisté CSS bez použití jakéhokoli frameworku je časově náročnější na tvorbu. To platí obzvlášť u rozsáhlých projektů, kde je zapotřebí použití komplexních stylizací. Bez systematického přístupu a pečlivé organizace se CSS kód stává rychle nepřehledným, těžko udržitelným a vývojář nad ním ztrácí kontrolu.

5.5 Různé připojení CSS stylů

Závěrečná fáze praktické části se zaměřuje na porovnání výkonu načítání webových stránek z jednoho CSS souboru pro jednu stránku, jednoho CSS souboru pro celý web a více CSS souboru uspořádaných dle komponentové metodiky. Analýza odhalila že načítání CSS z jednoho souboru pro konkrétní webovou stránku byla efektivnější než při komponentovém přístupu. Oproti tomu soubor který obsahoval CSS styly implementované pro celý web si vedl mnohem hůře.

Přístup s jedním souborem se styly vykazoval nejrychlejší metriky. To lze přisuzovat menšímu počtu HTTP požadavků pro zpracování, což zvyšuje celkovou dobu potřebnou k načítání. Implementace stylů pro celý web do jednoho souboru nám poukazuje na důležitost vyváženosti mezi konsolidací CSS stylů, zbytečným zatížením nepoužitelnými styly a opakujícími se styly.

Přesto, že se podařilo získat velké množství cenných informací a poznatků o efektivitě jednotlivých vlastností a nástrojů CSS, je důležité si uvědomit potencionální omezení a rozdíly, které s sebou mohou provedená testovací měření přinést, to nám poskytuje prostor k zamyšlení a diskusi nad možným směrem dalšího výzkumu.

Jedním z klíčových omezení našeho výzkumu je jeho provedení v podmínkách, které nemusí plně reflektovat složitost a dynamiku skutečných webových aplikací. Reálně jsou webové stránky zatíženy nejen CSS vlastnostmi, důležitou roli mají i externí skripty a multimediální obsah z různých zdrojů, což může mít zásadní vliv na rychlost načítání stránek. Dalším faktorem, který je třeba zvážit je softwarová a hardwarová vybavenost koncových uživatelů. Různý operační systém, odlišný prohlížeč a jiný typ zařízení, to jsou všechno skutečnosti, které mohou vést k odlišným výsledků a opět se nabízí možnost tyto vlivy v dalších měřeních a testováních zkoumat. V neposlední řadě je třeba zvážit vhodnou prioritizaci jednotlivých přístupů, kdy se můžeme dopracovat k tomu, že je v některých případech lepší, i na úkor nepatrného zhoršení výkonu, danou vlastnost nebo přístup zvolit, a to vzhledem k tomu, že tím dokážeme uživatelský prožitek a angažovanost uživatele několikanásobně zvýšit.

6 Závěr

Od začátku existence webových technologií, od prvních statických webových stránek tvořených převážně HTML kódem, až po interaktivní dynamické prostředí plné vizuální efektů mají vývojáři neustálé snahy najít rovnováhu mezi výkonností a vzhledem. Příchod CSS na scénu v devadesátých letech 20. století znamenal revoluci v oblasti webových technologií, kdy vznikl zcela nový obor, webový design, který získal zásadní postavení. Tento mezník znamenal nejen rozšíření kreativity ze strany vývojářů, ale zároveň zvýšení nároků a požadavků od uživatelů webu v podobě uživatelského prožitku. To dalo prostor pro vznik webové optimalizace, která si klade za cíl co nejvíce zefektivnit a optimalizovat webovou tvorbu. Jak se postupem času staly pojmy internet a web nezbytnou součástí každodenního života, tak se rychlost načítání webu, jeho konzistentnost a přístupnost staly nezbytnými standardy.

Cílem teoretické části této práce bylo vysvětlit a popsat tvorbu webových stránek a přiblížit uživatelům znalosti v oblasti nových HTML značek, CSS nástrojů a frameworků, měření rychlosti načítání webových stránek a jejich optimalizace. Tento cíl práce byl splněn, a to na základě použití odborné literatury dostupné v knižní i internetové podobě.

Cílem praktické části této práce bylo zrealizovat analýzu vlivu vlastností a nástrojů CSS na výkonnost odlišně složitých webových stránek. Ty byly vytvořeny za pomoci různých nástrojů, vlastností, přístupů a frameworků CSS, které jsou vhodné pro vznik optimalizovaného webu. Následně bylo provedeno zhodnocení výsledků a vyvození závěrečných doporučení, které nástroje, vlastnosti a frameworky CSS jsou rychlé, jaké mají největší vliv na rychlost načítání, které jsou přívětivé pro práci a které je tedy vhodné pro tvorbu webových stránek použít. Podařilo se zanalyzovat, jaký vliv na výkon mají různé layoutové techniky, detailně byly identifikovány vlastnosti CSS, které vykazují největší tendenci k ovlivnění výkonu. Dále se povedlo otestovat využívání frameworků, a to jak v experimentálním prostředí testovacích stránek, tak v prostředí reprezentativní stránky simulující realitu. Byly porovnány dva významné frameworky Tailwind a Bootstrap a byla zjištěna jejich škálovatelnost a vhodnost pro tvorbu stránek. Bylo zjištěno, jaký vliv má načítání CSS stylů z různě oddělených CSS souborů. Tímto došlo k naplnění cílů praktické části bakalářské práce.

Vykreslování webových stránek, pochopení kroků, jakými od počátečního požadavku uživatele až po zobrazení stránky celá stránka projde a co jednotlivé pojmy jako DOM, CSSOM nebo renderovací strom znamenají a zároveň se zorientovat v oblasti CSS Frameworků vyžaduje širokou míru explorace z dané problematiky, což jen potvrzuje komplexnost oblasti webové tvorby a webového designu. Za pomocí prostudování a představení způsobu měření

výkonnosti a optimalizace webu pomocí metrik Web Vitals, byl získán ucelený obraz o tom, jak jsou webové stránky hodnoceny, na co je třeba klást důraz a jaké mají hodnoty těchto metrik přímý dopad na koncové uživatele. I přes různé možnosti dnešního moderního CSS stále zůstává podstatné zachování základních principů a postupů psaní „čistého CSS“, které může v určitých případech vykazovat lepší hodnoty a jeho pochopení je nezbytné pro tvorbu optimalizovaného webu.

7 Seznam použitých zdrojů

ROUSE, Margaret. *Web*. Techopedia [online]. 2023. Dostupné z: https://www.techopedia.com/definition/5613/web

BERNERS LEE, Tim. *The World-Wide Web Initiative* [online]. In: Dostupné z: https://cds.cern.ch/record/2876913/files/930817.pdf

AWATI, Rahul. *World Wide Web (WWW)*. TechTarget [online]. 2023. Dostupné z: https://www.techtarget.com/ whatis/definition/World-Wide-Web.

ROSS, David. *How the web works*. MDN Web Docs [online]. 2023. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/How_the_Web_works?fbclid=IwAR1U- uiPtLXrrQTJotsNq4P_x%207aJCGZ9DZL_TT2bGvMnR-lLjlLB7i4J8Ss.

HIWARALE, Uday. *A brief overview of the TCP/IP model, SSL/TLS/HTTPS protocols and SSL certificates*. Medium [online]. 2020. Dostupné z: https://medium.com/jspoint/a-brief-overview-of-the-tcp-ip-model-ssl-tls-https-protocols-and-ssl-certificates-d5a6269fe29e

KASIREDDY, Preethi. *How the Web Works*. Freecodecamp [online]. 2015. Dostupné z: https://www.freecodecamp.org/news/how-the-web-works-a-primer-for-newcomers-to-web-development-or-anyone-really-b4584e63585c/?fbclid=IwAR3_lgf1iqrbNR3eP_QmkO%20SMU2X-whcccbvF042oLdTw-2tPFtCz8KkGBHk.

SIMMONS, Liz. *Front-End vs. Back-End*. Computerscience [online]. 2023. Dostupné z: https://www.computerscience.org/bootcamps/resources/frontend-vs-backend/

TANENBAUM, Andrew a David WETHERALL. *Computer Networks* [online]. 5th edition. Pearson, 2010. ISBN 978-0132126953. Dostupné z: https://learning.oreilly.com/library/view/computer-networks-fifth/9780133485936/

BODNAR, Danielle. *What Is a Web Browser?* Avast [online]. 2023. Dostupné z: https://www.avast.com/c-what-is-a-web-browser

GARSIEL, Tali. *How browsers work*. Web.dev [online]. 2011. Dostupné z: https://web.dev/articles/howbrowse rswork#render_tree_construction.

LENUKA, Melia. *Understanding the critical path.* WebDev [online]. 2023. Dostupné z: https://web.dev/learn/performance/understanding-the-critical-path?fbclid=IwAR0rAS6h7N4aWtKDb9DMAvC 6cq8wuKvgkvU_I-BqSfEyUTckgFaKfB1Q_-E

SPENCER, Corvin. *Critical rendering path*. Mdn web docs [online]. 2023. Dostupné z: https://developer.m ozilla.org/en-US/docs/Web/Performance/Critical_rendering_path.

HIWARALE, Uday. *How the browser renders a web page? — DOM, CSSOM, and Rendering*. Medium [online]. 2019. Dostupné z: https://medium.com/jspoint/how-the-browser-renders-a-web-page-dom-cssom-and-rendering-df10531c9969

GRIGORIK, Ilya. *Constructing the Object Model.* Web Dev [online]. 2014. Dostupné z: https://web.dev/articles/critical-rendering-path/constructing-the-object-model?fbclid=IwAR30SfjepR bMUaQ0FI2Ccc0xE8O_pgXmxOB1KAdPesLpn5uUbym-499Bpls

GRIGORIK, Ilya. Render-tree Construction, Layout, and Paint. Web dev [online]. 2023. Dostupné z: https://web.dev/articles/critical-rendering-path/render-tree-construction?fbclid= IwAR2dYMZ4TckMEia0SM CnTTeFM-i0jhJxNR6agimLw4EDXdqZHZf2I5Hhktc..

MEYER, Eric A. *CSS: the definitive guide: web layout and presentation* [online]. Fifth edition. Beijing: O'Reilly, 2023. ISBN 978-1098117610. Dostupné z: https://learning.oreilly.com/library/ view/css-the-definitive/9781098117603/

CASTRO, Elizabeth a HYSLOP, Bruce. In: *HTML5 a CSS3: názorný průvodce tvorbou WWW stránek*. Brno: Computer Press, 2012. ISBN 978-80-251-3733-8.

VANDEHEY, Scott. What is Modular CSS? Space Ninja [online]. 2018. Dostupné z: https://spaceninja.com/blog/2018/what-is-modular-css/

MICHÁLEK, Martin. *OOCSS: objektové psaní CSS*. Vzhurudolu [online]. 2017. Dostupné z: https://www.v zhurudolu.cz/prirucka/oocss..

MICHÁLEK, Martin. *BEM: Pojmenovávací konvence pro třídy v CSS*. Vzhurudolu [online]. 2017. Dostupné z: https://www.vzhurudolu.cz/prirucka/bem.

POLACEK, John. *Let's Define Exactly What Atomic CSS is*. Css-tricks [online]. 2017. Dostupné z: https://css-tricks.com/lets-define-exactly-atomic-css/.

LE, Duc. *Costly CSS Properties and How to Optimize Them*. *Dev* [online]. 2023. Dostupné z: https://dev.to/leduc1901/costly-css-properties-and-how-to-optimize-them-3bmd?fbclid= IwAR27oWGA7oFQ z7ko8-mFsyiDwF9piaC-IKw6dutBfNAEgEesbtxhsmA-Eq0

KARIMOV, Farda. *CSS Optimization: Avoiding and Optimizing for Faster Website Speed*. Medium [online]. 2023. Dostupné z: https://levelup.gitconnected.com/css-optimization-avoiding-and-optimizing-for-faster-website -speed-ccd33854cd01.

EDGAR, Matthew. *Speed Metrics Guide* [online]. Apress Berkeley, CA, 2024. Dostupné z: doi: https://doi.org/10.1007/979-8-8688-0155-6

WALTON, Philip. *User-centric performance metrics*. Web.dev [online]. 2019. Dostupné z: https://web.dev/art icles/user-centric-performancemetrics?fbclid=IwAR2f8lkc3Cx PLvSYVo29tiaXwPw7cgvi lpwKJCK7z1ucfgXF vkfJIIFi9ko#important-metrics-to-measure.

WALTON, Philip. *First Contentful Paint (FCP)*. Web.dev [online]. 2023. Dostupné z: https://web.dev/articl es/fcp.

WALTON, Philip a POLLARD, Barry. *Largest Contentful Paint (LCP)*. Web.dev [online]. 2023. Dostupné z: https://web.dev/articles/lcp.

WALTON, Philip. *First Input Delay (FID)*. Web.dev [online]. 2023. Dostupné z: https://web.dev/articles/fid.

POLLARD, Barry a WAGNER, Jeremy. *Interaction to Next Paint.* Web.dev [online]. 2023. Dostupné z: https://web.dev/articles/inp.

WALTON, Philip. *Time to Interactive (TTI)*. Web.dev [online]. 2023. Dostupné z: https://web.dev/articles/tti.

WAGNER, Jeremy a POLLARD, Barry. *Time to First Byte (TTFB)*. Web.dev [online]. 2023. Dostupné z: https://web.dev/articles/ttfb.

WALTON, Philip. *Total Blocking Time (TBT)*. Web.dev [online]. 2023. Dostupné z: https://web.dev/articles/tbt.

MIHAJLIJA, Milica a WALTON, Philip. *Cumulative Layout Shift (CLS)*. Web.dev [online]. 2023. Dostupné z: https://web.dev/articles/cls.

SENDPULSE. *Website Layout*. Sendpulse [online]. 2023. Dostupné z: https://sendpulse.com/support/glossary/websi te-layout.

THEWEBSITEARCHITECT. *Why is a layout important?* Thewebsitearchitect [online]. 2020. Dostupné z: https://the websitearchitect.com/why-is-the-layout-of-a-website-important/.

MDN Web Docs. *Float*. MDN Web Docs [online]. 2024 Dostupné z: https://developer.mozilla.org/en-US/docs/Web/CSS/float.

COYIER, Chris. *A Complete Guide to Flexbox*. Css-tricks [online]. 2022. Dostupné z: https://css-tricks.com/ snippets/css/a-guide-to-flexbox/#aa-background.

MICHÁLEK, Martin. In: *CSS: moderní layout*. [Praha]: Martin Michálek – Vzhůru dolů, [2022], ISBN 978-80-88253-07-5.

NIEDERST ROBBINS, Jennifer. *Learning web design: a beginner's guide to HTML, CSS, Javascript, and web graphics* [online]. Fifth edition. Beijing: O'Reilly, 2018. ISBN 978-1491960202. Dostupné z: https://learning.oreilly.com/library/view/learning-web-design/9781491960196/

MOZILLA, Developer. *Box-shadow*. Developer Mozilla [online]. 2024. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/CSS/box-shadow

MOZILLA, Developer. *Filter*. Developer Mozilla [online]. 2024. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/CSS/filter.

MOZILLA, Developer. *Transform*. Developer Mozilla [online]. 2024. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/CSS/transform.

Seznam obrázků, tabulek, grafů a zkratek

7.1 Seznam obrázků

7.2 Seznam tabulek

7.3 Seznam grafů

7.4 Seznam použitých zkratek

API – Application programming interface

CLS – Cumulative layout shift

CSS – Cascading styl sheets – Kaskádové styly

CSSOM – CSS object model

DNS – Domain name system – Systém pro vyhledávání domén

DOM – Document object model

FCP – First contentful paint

FID – First input delay

GUI – Graphical user interface – Grafické uživatelské rozhraní

HTML – Hypertext markup language – Značkovací jazyk pro tvorbu webových stránek

HTTP – Hypertext transfer protocol

INP – Interaction to next paint

IP – Internet protocol

LCP – Largest contentful paint

SEO – Search engine optimalization – Optimalizace pro vyhledávače

TBT – Total blocking time

TCP – Transmission control protocol

TTFB – Time to first byte

TTI – Time to interactive

UI – User interface – Uživatelské rozhraní

URL – Uniform resource locator

UX – User experience – Uživatelský prožitek

WWW – World wide web