Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Diplomová práce

Srovnání CSS frameworků podle rychlosti načtení webového obsahu

Bc. Tomáš Vébr

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Tomáš Vébr

Informatika

Název práce

Srovnání CSS frameworků podle rychlosti načtení webového obsahu

Název anglicky

Comparison of CSS frameworks according to loading speed of web content

---

Cíle práce

Hlavním cílem diplomové práce je porovnat vybrané CSS frameworky s obdobným charakterem využití se zaměřením na rychlost načtení webového obsahu.
Dílčími cíli jsou:
-Charakteristika využití CSS frameworků a technologií pro jejich vývoj
-Vytvořit přehled obdobných řešení a doporučení autorů zabývajících se obdobnou tématikou
-Vytvořit experimentální webové stránky s využitím vybraných frameworků
-Stanovení kritérií pro porovnání

Metodika

Teoretická část se bude zabývat využitím CSS frameworků při vývoji webových stránek. Popsány budou jednotlivé prvky používány k tvorbě stránek a dále technologie používány při vývoji frameworků. Dále budou představeny možná řešení, která budou předmětem srovnání.

V praktické části bude klíčovým úkolem navržení testovacích kritérií (metrik), která budou sloužit pro porovnání. Jednotlivá kritéria budou sledovat průběh načítání webového obsahu. Pro měření výsledků bude navrženo a následně vytvořeno několik variant webové stránky s použitím vybraných CSS frameworků. Pro sběr dat bude využito volně dostupných softwarových nástrojů, jakými jsou například Performance Insights a Lighthouse.

Na základě výsledků vlastního měření budou stanoveny závěry a doporučení.

Doporučený rozsah práce

60 – 80 stran (od Úvodu po Závěr)

Klíčová slova

CSS framework, rychlost načítání, doba načítání, porovnání

Doporučené zdroje informací

DI NOIA, Tommaso, In-Young KO, Markus SCHEDL a Carmelo ARDITO. Web Engineering. Berlin: Springer, 2022. ISBN 978-3-031-09916-8.

HOWE, Shay. LEARN TO CODE HTML & CSS: DEVELOP & STYLE WEBSITES. Ca: New Riders, 2014. ISBN 978-0-321-94052-0.

KING, Andrew. Website Optimization. California: O'Reilly, 2008. ISBN 9780596515089.

SHENOY, Aravind a Anirudh PRABHU. CSS Framework Alternatives: Explore Five Lightweight Alternatives to Bootstrap and Foundation with Project Examples. New York: Springer, 2018. ISBN 978-1-4842-3398-6.

SHIVAKUMAR, Shailesh Kumar. Modern Web Performance Optimization: Methods, Tools, and Patterns to Speed Up Digital Platforms. New York: Springer, 2020. ISBN 978-1-4842-6528-4.

Předběžný termín obhajoby

2023/24 LS – PEF

Vedoucí práce

Ing. Jan Masner, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 4. 7. 2023

doc. Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 3. 11. 2023

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 26. 03. 2024

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Srovnání CSS frameworků podle rychlosti načtení webového obsahu" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 31. 3. 2024 _____

Poděkování

Rád bych touto cestou poděkoval Ing. Janu Masnerovi, Ph.D. za vedené mé diplomové práce a poskytnutí podnětných nápadů na její vylepšení. Dále bych rád poděkoval svým rodičům za jejich podporu během celého studia. Poděkování patří také Bc. Petru Vršanovi za veškeré konzultace a poskytnuté rady.

Srovnání CSS frameworků podle rychlosti načtení webového obsahu

Abstrakt

Tato diplomová práce pojednává o CSS frameworcích a jejich vlivu na rychlost načítání webového obsahu. Hlavním cílem bylo srovnání vybraných CSS frameworků se stejným charakterem použití a zjištění jejich dopadu na rychlost načtení webového obsahu. Pro porovnání bylo zapotřebí nejprve navrhnout a následně pomocí každého frameworku vytvořit experimentální webové stránky. Samotné měření sledovaných metrik doplněných o váhy bylo provedeno nástrojem Lighthouse od společnosti Google. Z výsledků je patrné, že rozdíl mezi jednotlivými frameworky není zanedbatelný a může značně ovlivnit uživatelskou zkušenost. V závěru jsou formulovány konkrétní doporučení pro výběr testovaných frameworků s ohledem na co nejkratší dobu načítání webového obsahu.

Klíčová slova: CSS framework, rychlost načítání, doba načítání, porovnání, srovnání

Comparison of CSS frameworks according to loading speed of web content

Abstract

This thesis is about CSS frameworks and their impact on the loading speed of web content. The main objective was to compare selected CSS frameworks with the same usage pattern and determine their impact on the loading speed of web content. For the comparison, it was necessary to first design and then create experimental web pages using each framework. The actual measurement of the observed metrics, supplemented with weights, was performed with Google's Lighthouse tool. The results show that the difference between the frameworks is not negligible and can greatly affect the user experience. Finally, specific recommendations are formulated for the selection of the tested frameworks with respect to the shortest possible loading time for web content.

Keywords: CSS framework, loading speed, loading time, compare, comparison

Obsah

1 Úvod

Úspěšnost webových stránek je výrazně ovlivněna poskytovaným uživatelským zážitkem. Ke kvalitě uživatelského zážitku z používání webových stránek neodmyslitelně patří rychlost načítání obsahu. Uživatelé nejsou ochotni čekat déle nežli pár vteřin a v případě, že obsah stále není načtený stránky opouští. Stránky s dlouhou dobou načítání nejsou vyhledávacím algoritmem řazeny na předních příčkách. Každý majitel webových stránek se snaží zamezit nastání takové situace, jelikož tím dochází ke ztrátě potenciálních zákazníků.

Z tohoto důvodu je dobré upřít pozornost na technickou stránku webových stránek a prozkoumat všechny jejich aspekty. V případě optimalizace pouze určité oblasti nemusí dojít k požadovanému výsledku zlepšení, jelikož zbylé části neumožní provedené optimalizaci dostatečně vyniknout. Proto je důležité se zaměřit na všechny oblasti.

Jednou z oblastí je optimalizace kódu. V případě použití frameworků jsou však možnosti optimalizace výrazně omezené, a proto je třeba zvolit takový framework, který je již sám o sobě dobře optimalizovaný.

Tato práce se zabývá právě zkoumáním vybraných frameworků v oblasti rychlosti načtení webového obsahu. Cílem je poznání, jaký vliv na dobu načítání jednotlivé frameworky skutečně mají a dále poskytnutí uceleného pohledu na tuto problematiku vývojářům, kteří mohou tyto informace využít při výběru frameworku pro vlastní projekt.

2 Cíl práce a metodika

2.1 Cíl práce

Hlavním cílem diplomové práce je porovnat vybrané CSS frameworky s obdobným charakterem využití se zaměřením na rychlost načtení webového obsahu.

Dílčími cíli jsou:

-Charakteristika využití CSS frameworků a technologií pro jejich vývoj

-Vytvořit přehled obdobných řešení a doporučení autorů zabývajících se obdobnou tématikou

-Vytvořit experimentální webové stránky s využitím vybraných frameworků

-Stanovení kritérií pro porovnání

2.2 Metodika

Teoretická část se bude zabývat využitím CSS frameworků při vývoji webových stránek. Popsány budou jednotlivé prvky používány k tvorbě stránek a dále technologie používány při vývoji frameworků. Dále budou představena možná řešení, která budou předmětem srovnání.

V praktické části bude klíčovým úkolem navržení testovacích kritérií (metrik), která budou sloužit pro porovnání. Jednotlivá kritéria budou sledovat průběh načítání webového obsahu. Pro měření výsledků bude navrženo a následně vytvořeno několik variant webové stránky s použitím vybraných CSS frameworků. Pro sběr dat bude využito volně dostupných softwarových nástrojů, jakými jsou například Performance Insights a Lighthouse.

Na základě výsledků vlastního měření budou stanoveny závěry a doporučení.

3 Teoretická východiska

V této části se nachází teoretická východiska, která slouží jako podklad pro zpracování vlastní práce. Východiska pojednávají o základních informacích z oblasti vývoje webových stránek a postupně se více ponořují do oblasti pojednávající o rychlosti načítání webového obsahu.

3.1 HTML

HyperText Markup Language (zkráceně HTML) je představitelem takzvaných značkovacích jazyků. Jedná se o standardní značkovací jazyk pro tvorbu webových stránek. HTML definuje strukturu dokumentů a význam webového obsahu. (Harris, 2014)

Jak již bylo řečeno, HTML patří mezi značkovací, nikoli programovací jazyky. Není tedy možné vytvářet dynamické funkce. Momentálně je jazyk HTML považován za oficiální webový standard. World Wide Web Consortium (W3C) spravuje a nadále vyvíjí HTML specifikace spolu s poskytováním pravidelných aktualizací. (Harris, 2014)

3.1.1 HTML5

V současné době se nejnovější verze označuje jako HTML5. Ta byla vydána již koncem roku 2014 a představuje zásadní zlom oproti předchozím postupům značkování. Účelem těchto změn bylo standardizovat mnoho nových způsobů, které vývojáři používali. Současně byla snaha podpořit jednotnou sadu osvědčených postupů (tzv. „best practices“) s ohledem na vývoj webu. (Powell, 2010)

Většina jednotlivých změn je výsledkem větších cílů v návrhu jazyka. Mezi tyto cíle patří:

3.1.1.1 Podpora sémantického značení

Sémantické značení znamená značení, které má určitý význam nežli značení vyjadřující pouze vzhled daného prvku. Z tohoto důvodu bylo přidáno větší množství nových značek, které dokáží lépe specifikovat, o jaký obsah na stránce se jedná. (Powell, 2010)

Dříve bylo běžné uzavírat části stránky jakými jsou záhlaví, navigace, menu, postranní panel, hlavní obsah a patičku do blokového elementu . O jaký segment se jedná vypovídal primárně název přiřazené třídy. HTML5 přidává dedikované značky pro většinu takových případů (<section>, <article>, <header>, <footer>, <nav>, <aside>, <main>). (Powell, 2010)

3.1.1.2 Oddělení designu od obsahu

Současně se zaměřením na výrazné zlepšení sémantiky značek HTML5 reviduje starší značky. V případech, kdy značky neupravují význam obsahu, ale pouze předávají webovému prohlížeči informaci o tom, jak by měl daný obsah zobrazit přestávají být takové značky nadále podporovány. Pouze několik takových značek je nadále podporováno, avšak platí pro ně varování, že se obvykle nejedná o doporučený postup. (Howe, 2014)

3.1.1.3 Podpora přístupnosti a přizpůsobivosti designu

Z důvodu různorodé interakce s webem je ho zapotřebí upravit tak, aby s ním mohl pracovat kdokoli. Běžná zařízení, jakými jsou stolní počítače, laptopy, tablety a mobilní telefony představují velkou škálu velikostí obrazovek, poměrů stran a rozlišení. Již tato rozmanitost by sama o sobě měla stačit k podpoře sémantických a responzivních designových postupů. Ne každý uživatel však používá „běžný“ prohlížeč. (Wood, c2015-2023)

Především nevidomí a jinak zrakově postižení uživatelé nemohou procházet web běžným způsobem a jsou nuceni používat různé asistenční technologie. Čtečky obrazovky překládající obsah webu do mluvené podoby, tlumočníci Braillova písma, speciální nastavení prohlížeče pro zobrazování obsahu bez stylů nebo s vysokým kontrastem a navigace založená na klávesnici. Všem těmto technologiím brání značkování se snahou pevně zakódovat (tzv. „hard-code“) design a styl do obsahu stránky. (Harris, 2014)

3.1.1.4 Snížení překrývání mezi HTML, CSS a JavaScript

Vývoj webu z „front-end“ hlediska zajišťují 3 jazyky – HTML, CSS a JavaScript. V počátcích internetu nebylo specifikováno, co přesně má který jazyk zajišťovat, a tak se

díky paralelnímu vývoji v mnoha funkcionalitách a zaměřeních překrývaly. Bylo tedy zapotřebí definovat povahy a účely těchto jazyků a jejich omezení, případně rozšíření, aby dělaly to, co je v jejich povaze:

- HTML – Obsah
- CSS – Design
- JavaScript – Interakce

(Wood, c2015-2023)

3.1.1.5 Podpora multimédií bez nutnosti použití zásuvných modulů (tzv. „pluginů“) třetích stran (např. Flash nebo Java)

Při svém vzniku bylo HTML vytvořeno pro (hyper-)textové dokumenty obsahující maximálně několik málo obrázků, nikoliv však multimediální stránky se zvukem a videem. V momentě, kdy uživatelé začali přidávat takovýto druh obsahu na webové stránky, bylo vyžadováno přidávání speciálních zásuvných modulů (tzv. „pluginů“) třetích stran (např. Flash nebo Java) na klientské stanice. Tyto pluginy fungovaly špatně a způsobovaly bezpečnostní rizika. Základní funkce webových stránek musely být napsány v jiných jazycích a obsah stránek tak nebyl dostupný pro vyhledávače a čtečky obrazovky. HTML5 poskytuje podporu pro multimédia pomocí prvků <audio>, <video> a pro obrazce a grafiku vytvořenou pomocí JavaScriptu <canvas>. (Powell, 2010)

3.2 CSS

Kaskádové styly známé především pod zkratkou CSS (z anglického Cascading Style Sheets) byly vyvinuty World Wide Web konsorciem (zkráceně W3C) v roce 1996 z prostého důvodu. HTML elementy nebyly navrženy tak, aby obsahovaly značky (tzv. „tagy“), které by formátovaly vzhled stránky. (Powell, 2010)

V HTML verzi 3.2 byly představeny značky pro úpravu vzhledu, které však způsobily značné problémy pro vývojáře. Vzhledem k tomu, že webové stránky obsahují různé typy písem, barevná pozadí a další styly, přepisování kódu bylo zdlouhavé, špatně proveditelné a zbytečně nákladné. Z tohoto důvodu W3C vytvořilo jazyk CSS. (Powell, 2010)

Společně s myšlenkou HTML5 oddělit význam obsahu od jeho vzhledu je použití kaskádových stylů téměř podmínkou. Pro uživatele by nebyla webová stránka s použitím pouze HTML elementů atraktivní. (Harris, 2014)

CSS umožňuje nastavit veškeré stylování pro vytvoření požadovaného designu v samostatném souboru a až poté tento soubor připojit k HTML dokumentu. Díky tomu je samotné značkování v HTML souboru jednodušší, přehlednější a snadnější na údržbu. (Harris, 2014)

3.2.1 Výhody použití CSS

Stylování pomocí CSS poskytuje kromě již zmíněných výhod o snadnějším postupu stylování i mnohé další. Za zmínku stojí například:

3.2.1.1 Úspora času

Jedním z benefitů CSS je možnost použít vícekrát již vytvořené styly. Jeden soubor definující styly pro libovolný počet prvků může měnit vzhled mnoha stránek. Provedené změny jsou automaticky aplikovány na všechny dotčené elementy na všech stránkách, což vede ke snadnější údržbě. (Di Noia, 2022)

3.2.1.2 Rychlejší načítání

Oproti stylování pomocí HTML značek a atributů není potřeba mnohokrát opakovat ten samý kód. Stačí nadefinovat styl pro daný prvek jednou a odkazovat se na něho kolikrát je třeba. Zabraňuje se tak nadbytečnému množství kódu, což má za následek menší velikost souborů potřebnou pro stažení a tím rychlejší dobu načítání. (Di Noia, 2022)

3.2.1.3 Kvalitnější styly

V porovnání s HTML disponuje jazyk CSS mnohem širší paletou možností úprav jednotlivých elementů. Tím je tedy možné docílit mnohem hezčího a zajímavějšího vzhledu stránky. (Powell, 2010)

3.2.1.4 Kompatibilita

Kaskádové styly umožňují přizpůsobit zobrazovaný obsah pro více typů zobrazovacích zařízení. Zobrazení obsahu se přizpůsobuje zobrazovacímu zařízení, ať už podle rozlišení nebo poměru stran. Možnost zobrazovat obsah stejně kvalitně pro velké monitory stolních počítačů i malé obrazovky mobilních zařízení je klíčová. (Howe, 2014)

3.2.2 Stylování pomocí CSS

CSS používá jednoduchou syntaxi založenou na anglickém jazyce se sadou pravidel, kterými se řídí. Struktura CSS je velice jednoduchá. Je založena na selektoru a deklaračním bloku. Selektor označuje část HTML, která bude stylována. V deklaračním bloku se nachází samotné deklarace stylů odděleny středníky. Deklarace se skládá z CSS vlastnosti a hodnoty odděleny dvojtečkou. (Powell, 2010)

Výběr elementů určených ke stylování je určován selektory. Selektorů existuje několik typů. Nejčastějším selektorem je třída, kterou je možné přiřadit k libovolnému množství a typů elementů. Id selektor je unikátní a je ho možné přiřadit jen k jednomu konkrétnímu elementu. Dále lze provést selekci na základě použité značky elementu, styl se aplikuje pro všechny elementy se stejnou značkou. Posledním jednoduchým selektorem je globální selektor, který ovlivňuje všechny elementy na stránce. Konkretizovat výběr lze i kombinací zmíněných selektorů (typicky třída a značka) nebo použitím pseudotřídy nebo pseudoelementu. (Howe, 2014)

3.3 CSS framework

CSS framework je knihovna předem připravených CSS stylů. Kolekce šablon stylů usnadňuje práci vývojářům uživatelského rozhraní. Místo nutnosti začínat každý projekt zcela od začátku, frameworky poskytují nástroje pro rychlé vytvoření uživatelského rozhraní, které opětovně vytváří a vylepšují během zpracovávání projektu. Současně pomáhají vytvářet webové stránky více vyhovující stanoveným standardům. (Semah, 2023)

Frameworky jsou navrženy pro používání v typických případech, jakými jsou vývoj navigační lišty, patičky, hamburger menu, více sloupcové rozložení stránky a tak podobně. Dalším klíčovým benefitem kromě úspory času je usnadnění práce více lidí na projektu

díky standardům nabízených CSS frameworky. Díky předdefinovaným třídám musí všichni vývojáři používat stejné názvy tříd, je sjednoceno rozložení (tzv. „layout“) stránky, kód je dobře čitelný a snadno se v něm orientuje. Výsledkem je menší počet chyb a lepší týmová komunikace. (Adarsh, 2023)

3.3.1 Druhy CSS frameworků

CSS frameworky lze řadit do skupin podle vlastností a druhu použití na univerzální frameworky, užitkově založené frameworky, odlehčené frameworky, beztřídní frameworky a specializované frameworky. (Shenoy, 2018)

Univerzální frameworky jsou všestranné nástroje, které nabízejí širokou škálu předem navržených stylů, komponent a utilit pro vývoj webu. Tyto rámce jsou navrženy tak, aby vyhovovaly různým typům projektů a poskytovaly vývojářům komplexní sadu zdrojů pro zefektivnění procesu stylování. (Shenoy, 2018)

Užitkově založené CSS frameworky (utility-based), známe také jako funkční CSS frameworky jsou kategorií, která upřednostňuje jednoduchost, modularitu a efektivnost při vývoji front-endu. Tyto frameworky nabízejí jedinečný přístup ke stylování webových aplikací tím, že poskytují kolekci pomocných tříd, které lze kombinovat a vytvářet požadované styly rozvržení. Namísto spoléhání se na tradiční třídy CSS, které definují konkrétní styly pro jednotlivé prvky, utility-based frameworky nabízejí širokou škálu tříd, z nichž každá je zodpovědná za jednu vlastnost stylu. Tyto třídy lze snadno kombinovat v HTML značkách a dostáhnout tak požadovaného vzhledu. (Harnessing The Power Of Top 15 CSS Frameworks: Responsive Web Design Unleashed, 2023)

Odlehčené CSS frameworky představují odklon od tradičního objemu a složitosti spojené s rozsáhlými knihovnami stylů. Tyto frameworky upřednostňují efektivitu a minimalismus a nabízejí vývojářům štíhlou a agilní sadu nástrojů pro vytváření webového rozhraní. Namísto zahlcení vývojářů množstvím funkcí a komponent se lehké frameworky zaměřují na poskytování pouze základních stavebních bloků pro stylování. Tento přístup podporuje přímější spojení mezi vizí vývojáře a výsledným kódem. (Shenoy, 2018)

Beztřídní CSS frameworky nabízí odklon od tradičních přístupů ke stylování a rozvržení. Na rozdíl od svých protějšků s obrovským množstvím tříd tyto frameworky přijímají efektivnější a nenápadnější přístup ke stylování webových prvků. Tím, že

minimalizují závislost na CSS třídách, upřednostňují jednoduchost a efektivitu při vytváření webových rozhraní. Beztřídní frameworky spíše, než zahlcovat HTML kód četnými třídami povzbuzují vývojáře, aby k definování struktury stránky využívali sémantické značky HTML. Styl je poté aplikován přímo na tyto značky pomocí kontextových selektorů nebo jiných inovativních technik. Tento přístup nejen snižuje délku kódu, ale také zlepšuje jeho čitelnost a udržitelnost. (Harnessing The Power Of Top 15 CSS Frameworks: Responsive Web Design Unleashed, 2023)

Specializované CSS frameworky poskytují cílená řešení pro konkrétní úkoly a umožňují vývojářům v těchto doménách efektivně pracovat. Zatímco jejich rozsah může být ve srovnání s univerzálními frameworky omezený, vynikají v plnění jedinečných požadavků svých příslušných oblastí. (Harnessing The Power Of Top 15 CSS Frameworks: Responsive Web Design Unleashed, 2023)

3.3.2 Komponenty CSS frameworků

CSS frameworky jsou stejně jako jiné knihovny složeny z mnoha komponent. Které knihovny jsou v daném frameworku obsaženy závisí na vývojáři, avšak pro použitelnost frameworku jsou některé komponenty klíčové, tudíž se nachází prakticky ve všech případech. (Hernandez, 2023)

3.3.2.1 Struktury rozložení stránky

V každém frameworku se nachází struktury ke snadnější organizaci obsahu a designu stránky. Toho je docíleno dvěma způsoby. Častějším zástupcem je systém založen na komponentě „grid“. Jedná se o dvourozměrný systém založený na mřížkovém rozložení prvků. Díky svým vlastnostem nabízí širokou škálu možností, jak lze elementy na stránce rozmístit. Grid nabízí snadný způsob, jak centrovat prvky nejen horizontálně, ale i vertikálně. (CSS grid layout, 2023)

Druhým způsobem, jak vytvořit layout stránky je využít „flexbox“ komponentu. Ta oproti gridu operuje pouze v jedné dimenzi. Vybrat lze mezi orientací prvků horizontálně či vertikálně. Prvky také mohou měnit svoji velikost v závislosti na dostupném prostoru. (CSS flexible box layout, 2023)

3.3.2.2 Typografie

Text je jedním ze základních stavebních kamenů stránky. Většina informací je na stránkách v psané podobě a pro uživatele je klíčové, aby byl text dobře čitelný. Proto je nedílnou součástí vývoje stránek i stylování textu. Frameworky proto přichází s již přednastavenou typografií. Komplexní nastavení typografie obsahuje formátování fontu, barvy, velikosti písma, tloušťky, mezery mezi písmeny, slovy a řádky. (Abdaal, 2019)

3.3.2.3 Kompatibilita mezi prohlížeči

Kompatibilita napříč prohlížeči zajišťuje, že obsah bude zobrazen správně a konzistentně v jakémkoliv webovém prohlížeči. V případě, že se obsah zobrazuje chybně nebo části webové aplikace nefungují správně, uživatel nebude ochoten takové stránky navštěvovat. Proto je třeba při vývoji dbát na testování v mnoha prohlížečích. Při vývoji kvalitních frameworků byla kompatibilita zajištěna. (Gadhiya, 2023)

3.3.2.4 Pomocné třídy pro pozicování prvků

Pomocné třídy pomáhají předcházet opakování stejného kódu vytvářením sady abstraktních tříd, které mohou být opětovně používány na HTML elementy. Každá pomocná třída je zodpovědná za jeden úkol, který musí výborně plnit. Nejčastěji jsou pomocné třídy využívány pro: pozicování textu, pozicování bloků, definici šířky sloupců ve vícesloupcovém layoutu, margin a padding, definici šířky a výšky bloků, viditelnost prvků, hodnoty position a z-index, typografii, barvy, rámečky a hodnoty display. (Ajmi, 2014)

3.3.2.5 Užitkové třídy

Užitkové třídy neboli „utility-classes“ jsou samo-popisné, jednoúčelové CSS třídy. Vývojáři tyto třídy používají z důvodu snížení redundance CSS kódu. Často se opakující část kódu lze nadefinovat pouze jednou, ale použít v neomezeném množství. Díky názvům tříd vyjadřujících jejich chování je pro vývojáře snadnější orientace v HTML kódu, jelikož si mohou bez dalšího prohlížení kódu představit, jak se bude daný prvek chovat. (Bishop, 2019)

3.4 Technologie pro vývoj CSS frameworku

Při vývoji CSS frameworků se používá několik technologií. Hlavní komponentou jsou CSS preprocesory, které usnadňují psaní samotného CSS kódu. Dále je jako doplněk hojně používán programovací jazyk Javascript.

3.4.1 CSS preprocesory

Jazyk CSS má omezené možnosti, pokud jde o psaní logiky, organizaci kódu a provádění dalších výpočetních úkolů. Jako řešení těchto problémů byly vyvinuty CSS preprocesory. (Monus, 2023)

CSS preprocesory jsou skriptovací jazyky, které rozšiřují základní možnosti CSS. Umožňují používat logiku v CSS kódu, jako jsou proměnné, vnořování, dědičnost, funkce a matematické operace. Preprocesory usnadňují automatizaci opakujících se úloh, snižují počet chyb a velikost kódu, vytvářejí opakovatelně použitelné fragmenty kódu a zajišťují zpětnou kompatibilitu. (Monus, 2023)

Každý preprocesor má svou vlastní syntaxi. Pomocí kompilátorů je třeba vytvořit soubor v běžném CSS formátu, aby jej mohly webové prohlížeče vykreslit na straně klienta. Ačkoli dělají CSS preprocesory podobné věci, stále se od sebe více či méně odlišují. Kromě již zmiňované syntaxe jsou rozdíly v dostupných nástrojích, rámcích a knihovnách. (Monus, 2023)

Přestože se v posledních letech CSS výrazně vylepšilo díky zavedení vlastních a logických vlastností, matematických funkcí a funkcí pro deklaraci barev, nových pseudotříd a dalších vylepšení, stále existuje spoustu důvodů, proč preprocesory používat. Stále mohou pomoci s urychlením pracovního postupu, optimalizací základu kódu, vylepšením výkonu, přípravou škálování a dalšími vylepšeními. (Monus, 2023)

3.4.1.1 Sass

Sass (zkratka z anglického „Syntatically Awesome Style Sheets“) je nejstarší a nejrozšířenější CSS preprocesor. První verze vznikla již v roce 2006. Inspirací pro vznik byl jazyk Haml rozšiřující HTML o dynamické funkce. Prvotní snaha byla implementovat do CSS obdobné dynamické funkcionality. (Monus, 2023)

Sass rozšiřuje CSS o funkce z tradičních programovacích jazyků, zejména objektově orientovaných. Pro zápis je používán SassScript, což je dynamicky zapisovaný skriptovací jazyk. (Monus, 2023)

Původně byl Sass napsán v jazyce Ruby a bylo podmínkou, aby byl nainstalovaný i na zařízení, kde bude Sass používán. Syntaxe pro zápis se od CSS liší. Nepoužívají se složené závorky k ohraničení bloku kódu, namísto toho je kód pouze odsazen. Jednotlivé příkazy se oddělují pouhým zalomením řádku namísto obvyklého středníku. S klesající popularitou jazyka Ruby se stával i samotný Sass zastaralým. Z tohoto důvodu Sass představil nové implementace LibSass a Dart Sass. Obě tyto verze jsou rychlejší a snadno dostupné z npm[1]. (Monus, 2023)

LibSass je implementací Sass v jazycích C a C++. LibSass je dnes také považován za zastaralý, a proto je prosazován Dart Sass. Implementace Dart Sass je v jazyce Dart. Podmínkou pro jeho používání na systémech založených na JavaScriptu je integrace Dart Sass jako JavaScript knihovny. (SASS Vs SCSS: What's The Difference?, 2023)

Preprocesor Sass je neustále vyvíjen týmem skládajícím se z mnoha velkých technologických organizací a nezávislých vývojářů a je považován za standard ve vývoji webových aplikací na komerční úrovni. Je organizacemi preferovaným jazykem rozšiřující CSS pro vývoj složitých webových projektů. (Monus, 2023)

Klíčové vlastnosti

Jednou z hlavních vlastností Sass je vysoká kompatibilita s kaskádovými styly, což znamená, že Sass poskytuje podporu pro mnoho verzí CSS. Sass je kompatibilní se všemi verzemi CSS. (Sass, c2006-2023)

Sass proměnné mají tzv. „scope", mohou tak být používány lokálně i globálně. To dodržuje DRY[2] princip programování. K aplikaci DRY principu slouží dvě vlastnosti, a to mixiny a pravidlo @extend. (Sass, c2006-2023)

---

[1] Npm je správce balíčků pro jazyk JavaScript běžící na platformě Node.js. Umožňuje správu a sdílení knihoven, které lze nainstalovat do projektů.

[2] DRY (zkratka z anglického „Don't Repeat Yourself") je princip v programování, snažící se eliminovat duplicitu kódu.

Mixiny umožňují vytvářet skupiny souvisejících CSS pravidel a aplikovat je na libovolnou vlastnost. Pravidlo @extend přidává do jazyka Sass dědičnost. Své uplatnění nachází například, pokud existuje více různých designových prvků, které sdílí určité vlastnosti. Pomocí pravidla @extend je možné přidávat vlastnosti libovolné třídy do jiné. (Sass, c2006-2023)

Sass umožňuje také vnořování, což zlepšuje čitelnost kódu, který se navíc snáze udržuje. Lze jej použít při práci s CSS selektory, které sdílejí stejného rodiče. (Sass, c2006-2023)

Cykly a podmínky jsou velkou předností Sass, jelikož umožňují psát CSS pravidla stejně jako v jakémkoliv skriptovacím jazyce. Sass má vestavěná pravidla @if a @else, která umožňují testovat podmínky a také @for, @each a @while cykly. (Sass, c2006-2023)

Sass podporuje modularitu také prostřednictvím dílčích Sass souborů, které obsahují menší bloky kódu, které mohou být použity vícekrát. Dílčí soubory se identifikují přidáním podtržítka na začátek názvu souboru. Obsah dílčího souboru lze přidat do jiného Sass souboru pomocí pravidla @import. (Sass, c2006-2023)

Preprocesor Sass má také vestavěné funkce pro převod a míchání barev, manipulace s řetězci, provádění matematických výpočtů a dalších dynamických funkcí k tvorbě designu. Kromě toho lze definovat vlastní funkce. (Sass, c2006-2023)

3.4.1.2 SCSS

Sass disponuje dvěma syntaxemi. Původní s odsazením a novější Scss (celým názvem „Sassy CSS“). Tato syntaxe se svým formátováním podobá klasickému CSS. Blok kódu je označen složenými závorkami a jednotlivá pravidla jsou oddělována středníkem. Díky tomu je syntaxe Scss přísnější, ale zároveň výraznější z hlediska čitelnosti. Scss je postaveno na CSS a obsahuje další funkce kromě všech funkcí CSS. Scss je tedy nadmnožinou CSS. Hlavním cílem Scss je vyplnit mezery a nekompatibility mezi CSS a Sass. (SASS Vs SCSS: What’s The Difference?, 2023)

Klíčové vlastnosti

Ačkoliv je Scss pouze jinou syntaxí pro použití Sass a sdílí tak většinu vlastností, Scss nabízí několik vylepšení oproti Sass. Scss je nadmnožinou CSS, a proto vše, co lze

dělat s klasickým CSS lze i s Scss (na rozdíl od Sass). Dále umožňuje vkládání komentáře přímo do kódu. Svoji výraznější syntaxí navíc přidává na přehlednosti a zvyšuje míru kontroly nad kódem. (SASS Vs SCSS: What's The Difference?, 2023)

3.4.1.3 LESS

Less (zkratka z anglického „Leaner Style Sheets") byl vydán v roce 2009, 3 roky po vydání preprocesoru Sass. Samotný vývoj Sass značně ovlivnil. Less do sebe implementoval množství vlastností, kterými disponuje Sass (např. vnořování, proměnné, mixiny). Naopak Less ovlivnil Sass svou syntaxí, kterou se Sass inspiroval a vytvořil druhou syntaxi pojmenovanou Scss. (Monus, 2023)

Preprocesor Less je JavaScript knihovna rozšiřující funkcionality běžného CSS. Jelikož je Less napsán v jazyce JavaScript, lze jej spustit jako balíček npm, nebo zkompilovat Less přímo v prohlížeči přidáním souborů s příponou .less a konvertoru Less do sekce v HTML dokumentu.

Klíčové vlastnosti

Stejně jako Sass proměnné, i Less proměnné mají tzv. „scope", který umožňuje rozhodnout, kde se budou proměnné volat. Less proměnné lze použít v CSS pravidlech, názvech selektorů a vlastností, URL adresách a příkazech @import. (Less, c2023)

Less mixiny fungují obdobně jako mixiny pro Sass. Lze definovat sadu souvisejících pravidel stylu a znovu je použít v celém kódu. (Less, c2023)

Preprocesor Less disponuje také specifickými „guarded mixiny", které implementují základní podmíněnou logiku v Less. Tato funkce je důležitá, jelikož Less nemá natolik pokročilou podmíněnou logiku jako Sass (např. nemá příkaz if-else). Guarded mixiny alespoň částečně kompenzují tento nedostatek. (Less, c2023)

Umožněn je i přístup k vestavěným funkcím, pomocí nichž lze manipulovat s barvami, obrázky, přechody, rozměry, jednotkami a dalšími funkcemi.

Princip DRY a dědičnost jsou také nedílnou součástí syntaxe Less. Pseudotřídu :extend lze použít k rozšíření selektorů a funkci sloučení k agregaci hodnot z více vlastností. Less také podporuje vnořování, takže je umožněno docílit čitelného a přehledného kódu. (Less, c2023)

3.4.1.4 Stylus

První verze preprocesoru Stylus byla vydána v roce 2010. Stylus je napsán v jazyce JavaScript, a tak ho mohou vývojáři jednoduše integrovat do Node.js projektů. Vývoj Stylusu ovlivnily Sass i Less. Oba preprocesory kombinuje a přejímá jejich přednosti (Sass logiku a Less jednoduchost a přímočarost). (Monus, 2023)

Stylus je mezi vývojáři oblíbený díky své stručné a flexibilní syntaxi. Kód se zapisuje do souborů s příponou .styl a umožňuje provádět zápis množstvím různých způsobů. Lze použít standardní syntaxe CSS, ale také lze zapisovat s vynecháním závorek, dvojteček a středníků nebo vynecháním veškeré interpunkce. (Monus, 2023)

Klíčové vlastnosti

Stylus mixiny fungují obdobně jako mixiny pro Sass a Less. Lze definovat sadu souvisejících pravidel stylu a znovu je použít v celém kódu. Jedinečnou funkcí jsou však transparentní mixiny. Ty umožňují automatické přidávání prefixů vývojářů k novějším CSS vlastnostem, které doposud nemají dostatečnou podporu webových prohlížečů. Stylus také automaticky přidává prefixy vývojářů ke klíčovým snímkům, tudíž je ideálním CSS preprocesorem pro vytváření @keyframe animací. (Monus, 2023)

Stylus má několik vestavěných funkcí pro manipulaci a převod barev a jednotek, výpočet průměrných, minimálních a maximálních hodnot, shodu vzorů a provádění dalších akcí. Podobně jako Sass umí i Stylus vytvářet vlastní funkce. (Monus, 2023)

Součástí preprocesoru je i podmíněná logika. Obsahuje podmínky if, else, else if a unless. Poslední zmíněný unless je logickým opakem if. Slovním vyjádřením může být „pokud ne“. Pomocí Stylusu lze také využít podmínky postfixu a cyklu. (Monus, 2023)

Dalšími pokročilými funkcemi jsou vyhledávání vlastností, které umožňuje odkazovat na předchozí vlastnosti bez jejich přiřazení k proměnným a částečný odkaz, který umožňuje přístup pouze k určitému počtu úrovní vnořených selektorů. (Monus, 2023)

3.4.2 JavaScript

JavaScript se současně řadí mezi skriptovací a programovací jazyky. Javascript se stará o dynamické prvky a interakci s uživatelem při vývoji webových stránek, aplikací nebo her. Především je JavaScript využíván společně s HTML a CSS. Skriptovací jazyk

dobře spolupracuje s CSS při formátování HTML prvků. JavaScript však stále udržuje interakci s uživatelem, což CSS samo o sobě nedokáže. (What is JavaScript?, 2023)

3.4.2.1 Běh kódu

Javascript může být spouštěn na straně klienta (client-side) i na straně serveru (server-side). Server-side JavaScript umožňuje back-end přístup do databází, souborových systémů a serverů. Kód je spouštěn na serveru ještě před tím, než je jakýkoliv obsah odeslán do klientského prohlížeče. Existuje několik technologií a frameworků, které umožňují použití JavaScriptu na serveru, příkladem může být Node.js. Tyto technologie nabízí prostředí pro běh programu, které umožňují serveru spouštět kód. JavaScript na straně serveru je používán k vytváření škálovatelných, vysoce výkonných webových aplikací, které zpracovávají velké množství dat a provoz. Vytváří také rozhraní API (zkratka z anglického „Application Programming Interface“), která umožňují aplikacím na straně klienta komunikovat se serverem. JavaScript kód je před klienty skryt, což znamená, že z klienta není kód viditelný ani jinak přístupný. Toto je důležitý aspekt v otázce bezpečnosti a ochrany citlivých údajů. (Batool, 2023)

V client-side JavaScriptu je skript stahován do klientské stanice a zpracováván ve webovém prohlížeči klienta. JavaScript kód je vložen přímo do těla webové stránky, nebo je odkazován prostřednictvím samostatného souboru s příponou .js. Když koncový uživatel navštíví stránku obsahující JavaScript, webový prohlížeč spustí skript společně s HTML a CSS kódem. Webový prohlížeč zpracovává kód postupně shora dolů, je tedy třeba deklarovat objekty a proměnné dříve, než je s nimi vykonána operace. JavaScript kód má v prostředí prohlížeče přístup k věcem poskytovaným pouze prohlížečem, jakými je tělo dokumentu aktuální stránky, okno, vyskakovací zprávy a podobně. Client-side Javascript je používán k ověřování vstupů, manipulací s prvky uživatelského rozhraní, přidávání animací, vyskakovacích oken, vyhledávacích polí, tlačítek, multimediálního obsahu, chatovacích widgetů a dalších interaktivních prvků webu. Jednoduše řečeno, JavaScript poskytuje možnost přidat na stránky logiku a dynamicky měnit obsah v závislosti na vstupech. Doplňuje tak jazyky HTML a CSS a umožňuje vytvářet obsáhlejší a zajímavější weby. (Batool, 2023)

3.4.2.2 JavaScript v CSS frameworku

Některé CSS frameworky obsahují JavaScript knihovny obsahující kolekce předem napsaného kódu. Tyto komponenty využívají JavaScript k aktivaci určitých interaktivních funkcí a chování. Příkladem komponent jsou rozbalovací nabídky, karusely, modály, popisky a další. Využívání CSS frameworků s vestavěnými JavaScript komponentami poskytuje vývojářům výhodu. Vývojáři tak mohou vytvářet interaktivní prvky bez nutnosti psaní kódu, ve spoustě případů stačí jen použít správnou knihovnu, případně kód mírně upravit dle potřeby. (Shivakumar, 2020)

Ačkoli CSS frameworky s vestavěnými JavaScript komponentami zrychlují vývoj stránek, mohou také přidávat určitou režii k načítání stránky a výkonu. Proto je klíčové posoudit potřeby projektu a použít pouze nezbytné komponenty, aby byl zajištěn optimální uživatelský prožitek. Současně je nezbytné udržovat frameworky třetích stran aktualizované, aby mohlo docházet k opravám chyb, bezpečnostním aktualizacím a přidáváním nových funkcí. V případě použití více knihoven v jednom projektu hrozí potenciální výskyt konfliktů mezi nimi. (Bin Uzayr, 2022)

3.5 Úprava CSS frameworku

CSS frameworky jsou populárními nástroji pro front-end vývoj, jelikož poskytují již hotové komponenty, rozložení stránek a styly, které šetří čas a zajišťují konzistenci. Přesto může nastat situace, kdy framework nevyhovuje specifickým potřebám a je třeba ho modifikovat nebo rozšířit bez poškození funkcionalit a kompatibility. (Amollo, 2023)

Jedním ze způsobů je využití Sass proměnných, které fungují jako zástupné symboly (tzv. „placeholdery) k hodnotám, které mohou být opětovně použity a změněny v celém kódu. Vytvořením vlastního souboru s příponou .scss lze hodnoty proměnných přepsat, tím změnit chování frameworku ale současně ponechat původní zdrojový kód. (Amollo, 2023)

Další možností úpravy a rozšíření frameworku je vytvoření a použití vlastních tříd. Díky použití vlastních tříd je možné přidat nebo přepsat styly, které nejsou frameworkem poskytovány nebo kolidují s vaším designem. (Amollo, 2023)

Více pokročilým způsobem pro rozšíření a přizpůsobení frameworku je využití mixinů a funkcí. Jedná se o bloky kódu, které lze opakovaně použít a vyvolat s různými

argumenty. Mixiny a funkce mohou pomoci s vytvořením komplexních stylů a logiky, které nejsou obsaženy ve frameworku nebo které vyžadují velkou míru opakování. (Amollo, 2023)

Posledním způsobem přizpůsobení a rozšíření frameworku je použití dědičnosti a specifičnosti, což jsou koncepty, které určují, jak jsou CSS pravidla aplikována a přepisována. Dědičností se rozumí, že jsou některé vlastnosti předávány z nadřazených prvků na prvky podřízené, pokud nejsou explicitně přepsány. Specifičnost pak znamená, že některé selektory mají vyšší váhu než jiné v závislosti na jejich typu, pořadí a úrovni podrobností. Použitím dědičnosti a specifičnosti lze upravit a vylepšit styly komponent frameworku pomocí vlastností jako „inherit“, „initial“ nebo „unset" nebo pomocí selektorů, jakými jsou například „:not“, „:hover“ nebo „:nth-child“. Při jejich použití je třeba dbát, aby nedocházelo k vytváření konfliktů nebo zmatků s pravidly frameworku a také aby nedocházelo k vytvoření příliš komplexního a obtížně udržovatelného kódu. (Amollo, 2023)

3.6 Konkrétní příklady CSS frameworků

Následující výčet představuje nejpopulárnější CSS frameworky pro univerzální použití.

3.6.1 Bootstrap

Bootstrap je jedním z nejpopulárnějších CSS frameworků. Svědčí o tom přibližně 5,5 milionu stažení za týden. Bootstrap byl původně vytvořen společností Twitter pro podporu konzistence mezi interními nástroji. V roce 2011 byl pak vydán jako otevřený software. Nyní je dostupná nejnovější verze 5.3.3. (Adarsh, 2023)

Bootstrap poskytuje responzivní design a jako jeden z prvních začal prosazovat mobile-first přístup. Poskytl správné nástroje pro jeho implementaci skrze zavedení mřížky (tzv. „gird“), která rozděluje obrazovku na sloupce. (Adarsh, 2023)

3.6.2 Foundation

Foundation je jedním z nejvíce pokročilých frameworků pro tvorbu uživatelského rozhraní. Přesto jeho popularita klesá a počet stažení za týden se pohybuje kolem hranice

jednoho tisíce. Vývoj zajišťuje společnost ZURB a první verze byla vydána v roce 2011. Dnes je nejnovějším vydáním verze 6.8.1. Důvodem pro vznik byla snaha o zvýšení efektivnosti při vývoji webových stránek. Foundation disponuje všemi prvky pro tvorbu responzivních a dobře vypadajících aplikací. Stejně jako většina frameworků zakládá responzivitu na mřížkovém systému. (Adarsh, 2023)

3.6.3 Bulma

Bulma je responzivní CSS framework založený na flexboxu. První verze byla vydána v roce 2016 a ihned se Bulma těšila velké popularitě. Nejnovější verze 0.9.4 dosahuje okolo 250 tisíc stažení za týden. Bulma je vytvořena pouze za pomoci CSS, avšak JavaScript kód je součástí dokumentace u daného prvku, takže i méně zkušený vývojář bez větších problémů zvládne tyto prvky implementovat. (Adarsh, 2023)

3.6.4 Fomantic UI

Fomantic UI je oficiální komunitní verze z původního, dnes již nevyvíjeného frameworku Semantic UI. Fomantic UI je navržen tak, aby byl nejen uživatelsky přívětivý, ale také se zaměřuje na použití sémantického HTML a efektivně podporuje přístupnost pro uživatele s omezením – WAI-ARIA. Nejnovější dostupná verze je 2.9.3 a Fomantic UI dosahuje hodnoty 14 tisíc stažení za týden. (Adarsh, 2023)

3.6.5 Blaze UI

Blaze UI svou popularitou nedosahuje takových rozměrů jako jiné frameworky, přesto si své místo najde. Počet stažení za týden dosahuje pouze přibližně jednoho tisíce. Celý framework je založen na mřížkovém systému. Od ostatních frameworků se liší používáním vlastních značek pro některé prvky, což není úplně obvyklé, nicméně není v rozporu se správnou sémantikou HTML. (Adarsh, 2023)

3.7 Prvky pro stavbu webových stránek

Webové stránky mohou vypadat navzájem velmi odlišně, avšak všechny mají tendenci sdílet podobné standardní komponenty. Mezi základní blokové elementy

definující větší oblast stránky patří hlavička, navigační lišta, hlavní obsah, postranní panel a patička. (Document and website structure, 2023)

Hlavička bývá obvykle pruh přes horní část vyplněný velkým nadpisem, logem či případně sloganem. Hlavička zůstává zpravidla neměnná napříč všemi stránkami konkrétního webu. (Document and website structure, 2023)

Navigace má za úkol odkazovat na hlavní sekce webu. Pro navigaci je důležitá konzistence napříč webem, aby byla pro uživatele přehledná. Pro neustálou dostupnost navigace při procházení stránek je využíváno jejího přichycení na vrchní části obrazovky. Pro mobilní zobrazení je pak navigace zprostředkována přes takzvané „hamburger menu“, které se rozbalí po stisku tlačítka. Ačkoli je navigační panel považován mnoha webovými designéry spíše za součást hlavičky než za skutečnou komponentu, není to podmínkou. Naopak existuje tvrzení, které doporučuje dva oddělené prvky, jelikož jsou tak lépe čitelné pro čtečky obrazovky, a tedy i pro přístupnost. (Document and website structure, 2023), (Hill, 2023)

Hlavní obsah je velká oblast uprostřed stránky, která obsahuje většinu jedinečného obsahu dané webové stránky, například video, které chce uživatel sledovat, hlavní příběh, který čte, nebo mapu, kterou chce zobrazit. Toto je část webu, která se zcela určitě liší stránku od stránky. (Document and website structure, 2023)

Postranní panel je používán k zobrazování periferních informací, odkazů, citací, reklam atd. Obvykle existuje propojení s hlavním obsahem, ať už se jedná o doplňkové informace nebo odkazy na související obsah, ale také se postranního panelu využívá pro sekundární navigační systém. (Document and website structure, 2023)

Patička bývá obvykle pruh ve spodní části stránky a obsahuje upozornění na autorská práva nebo kontaktní údaje. Je to místo pro vložení běžných informací, které nesou spíše informace o stránce nežli spojitost se samotným obsahem. (Document and website structure, 2023), (Hill, 2023)

Tyto hlavní komponenty tvoří základní rozdělení stránky a jsou použity defacto vždy. Další prvky jsou více volitelné a jejich použití závisí spíše na typu stránky nebo preferencích autora. (Friedman, 2018)

Hero section neboli v překladu „sekce hrdiny“ je oblast v horní části stránky (vždy nad místem „ohybu“, tedy kde se musí pro zobrazení dalšího obsahu posouvat) představující silný vizuální poutač. Obvykle bývá jejím obsahem obrázek, posuvník,

chytlavý kus typografie, video nebo cokoliv jiného, co přitahuje pozornost a předává potřebnou zprávu. Motivem fotografie nemusí být člověk, zvíře ani maskot. Obsahem může být produktová fotografie, obrázek s motivem krajiny nebo dokonce abstrakce. Hlavní myšlenkou je upoutat pozornost uživatele a navázání vizuálního, emocionálního a informativního spojení, zapojení je do posouvání stránky nebo mačkání tlačítek, aby se dozvěděli více. (Yalanska, 2023)

Slider neboli posuvník je interaktivní prvek využívající techniku slideshow nebo karuselu k prezentaci různých produktů, nabídek apod. Oblíbený je zejména jako součást e-commerce a firemních webů k prezentaci jakési galerie produktů nebo služeb. (Yalanska, 2023)

Vyhledávání je funkcionalita umožňující návštěvníkovi procházet obsah na webu a zobrazovat jej podle vyhledávacího dotazu. Správně nastavené vyhledávání zobrazuje relevantní obsah a poskytuje zkratku k vyhledávanému cíli. Interní vyhledávání tak šetří čas a úsilí uživatele, zvyšuje použitelnost, pomáhá udržet uživatele na stránce a zvyšuje míru konverze. Interaktivním prvkem odpovědným za vnitřní vyhledávání v uživatelském rozhraní je vyhledávací pole umožňující uživateli zadávat vyhledávací dotazy. Vyhledávání by však nemělo být upřednostňováno před běžnou navigací, jelikož ne všichni uživatele mají jazykové schopnosti pro vytvoření správného dotazu. Také není třeba vyhledávacího pole na webech o rozsahu maximálně pár stránek se spíše stručným obsahem. (Yalanska, 2023)

Breadcrumbs neboli drobečková navigace jsou navigační prvky používané k podpoře uživatelů při procházení webu. Uživatelé si lépe uvědomují, kde se na webu nacházejí a lépe si zvykají na strukturu webu. Drobečková navigace představuje sekundární úroveň navigace a zvyšuje použitelnost webu v případě, že se skládá z mnoha stránek. (Hill, 2023)

Formulář je interaktivní prvek, který umožňuje uživatelům odesílat informace do systému nebo serveru. Formuláře by měly být velmi jednoduché a snadno použitelné. Čím jednodušší by měl prvek být, tím více času by mělo být věnováno k jeho návrhu. Formulář by měl mít intuitivní navigaci a minimální počet požadovaných akcí. Ideálně je formulář vybaven vizuálními výzvami a tipy na podporu uživatele v procesu vyplňování. (Yalanska, 2023)

Karty, někdy také nazývány dlaždice jsou prvky rozvržení, které pomáhají uspořádat a vizualizovat homogenní data nebo obsah skenovatelným a snadno použitelným způsobem. Karty jsou obvykle uspořádány do jakési mřížky, ale každá karta vypadá v tomto systému jako samostatný díl. Karty mohou kombinovat různé typy obsahu o konkrétní položce. (Yalanska, 2023)

Video není úplně základní součástí webové stránky, ale s pokrokem řešení pro vývoj webu a technickými schopnostmi je v dnešní době na různých webech najdeme stále častěji. Video je ideálním prvkem pro upoutání pozornosti uživatele, jelikož působí na více kanálů vnímání (sluch, zrak a pohyb) současně a se jedná o prověřenou metodu, jak uživatele rychle a srozumitelně informovat. I proto jsou videa často součástí Hero sekcí. Vkládání videí do obsahu stránek má i svá úskalí (doba načítání, problém s kontrastem, odezva), které mohou kazit uživatelský dojem, a proto je třeba tento prvek používat s rozvahou. (Yalanska, 2023)

CTA tlačítko (call-to-action), neboli tlačítko s výzvou k akci je prvkem uživatelského rozhraní, jehož cílem je povzbudit uživatele k provedení určité akce. Tato akce představuje konverzi pro konkrétní stránku nebo obrazovku (např. nákup, kontakt, odběr atd.). Technicky se může jednat o jakýkoliv typ tlačítka, který je podporován textem výzvy k akci. Tento typ tlačítka se liší od všech ostatních tlačítek na stránce nebo obrazovce díky své poutavé povaze, která musí upoutat pozornost a stimulovat uživatele k provedení požadované akce. (Hill, 2023)

Modal označuje vyskakovací okno, které může nabývat různých podob. Často bývají vyskakovací okna používána pro přihlášení a registraci, možnosti využití jsou ovšem mnohem rozsáhlejší a obsahem tak může být téměř cokoli. (Hill, 2023)

3.8 Responzivita a Mobile-first přístup

S příchodem mobilních zařízení podporujících procházení webu bylo třeba přizpůsobit jeho vývoj. Primárně bylo třeba zajistit, aby se zobrazovaný obsah přizpůsoboval velikosti zobrazovacího zařízení. (Karlsson, 2014)

3.8.1 Responzivita

Touto problematikou se zabývá responzivita. Jedná se o přístup, který zajišťuje dobré vykreslení obsahu na všech různých zařízeních nehledě na velikost a rozlišení obrazovky. Současně zajišťuje dobrou použitelnost takových stránek.

S příchodem responzivního designu byly hojně využívány Media Queries. Ty umožňují nastavit tzv. „breakpointy“, tedy hranice, kdy dochází ke změně ve stylu. Tato hranice je nejčastěji reprezentována šířkou obrazovky v jednotkách pixel. Pro každou hranici lze styly upravit a dosáhnout tak lepšího zobrazení. (Friedman, 2018)

Media Queries jsou stále hojně využívána, avšak primárním způsobem pro pozicování elementů na stránce se stalo využití flexboxu a gridu. Ty jsou sami o sobě responzivní a dokáží se tak automaticky přizpůsobit velikosti obrazovky. Media Queries jsou tak využívána spíše pro změnu velikosti písma a změnu dalších grafických prvků. (Friedman, 2018)

3.8.2 Mobile-first přístup

Přestože responzivním designem lze docílit přizpůsobení obsahu podle velikosti zobrazovacího zařízení, je třeba navíc zajistit i specifické potřeby pro uživatele přistupujících na webové stránky z mobilního zařízení. (Unadkat, 2023)

Při použití mobile-first přístupu obecně platí, že vývoj webových stránek začíná vývojem stránek pro mobilní zařízení a až následně je zaměřována pozornost na větší zařízení. Společně s tím platí, že uživatelé mobilních zařízení mají jiné potřeby. Ovládání webu musí být přizpůsobeno pro dotykové obrazovky a prvky na stránce rozmístěny tak, aby byly snadno dosažitelné. Zároveň důvod návštěvy webové stránky na mobilním zařízení se může lišit od důvodu návštěvy skrze stolní počítač. Příkladem může být návštěva webových stránek restauračního zařízení, kdy uživatele přistupujícího na stránky skrze mobilní telefon pravděpodobně zajímá otevírací doba, kontakt či jídelní lístek a nikoliv galerie apod. Při vývoji je třeba přemýšlet nad důvodem návštěvy stránek z jakého typu zařízení a podle toho stránky přizpůsobit. (Bin Uzayr, 2022)

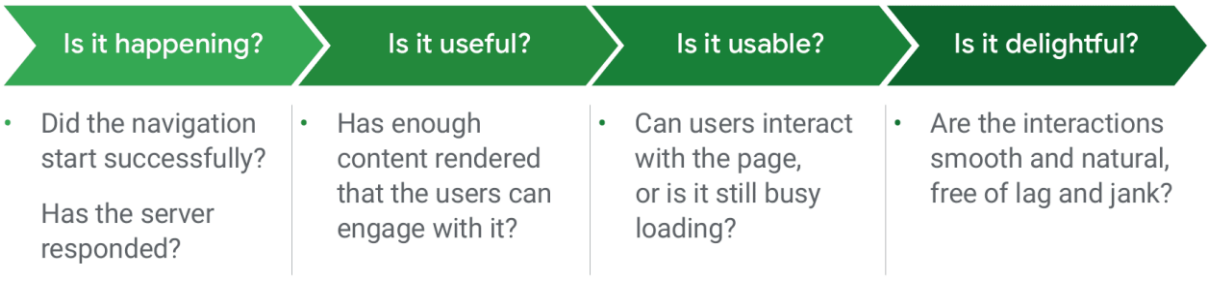
3.9 Metriky výkonu zaměřené na uživatele

Při vývoji moderního webového prostředí je klíčové měřit, optimalizovat a monitorovat, pokud mají stránky být a nadále zůstat rychlé. Výkon hraje významnou roli v úspěchu jakéhokoliv online podniku, protože vysoce výkonné weby zaujmou a udrží koncové uživatele lépe než ty se špatným výkonem. Weby by se měly zaměřit na optimalizaci pro metriky spokojenosti zaměřené na uživatele. Nástroje jako Lighthouse tyto metriky zvýrazňují a dopomáhají k podnikání správných kroků ke zlepšení výkonu. (Fast load times, c2023)

3.9.1 Rychlost

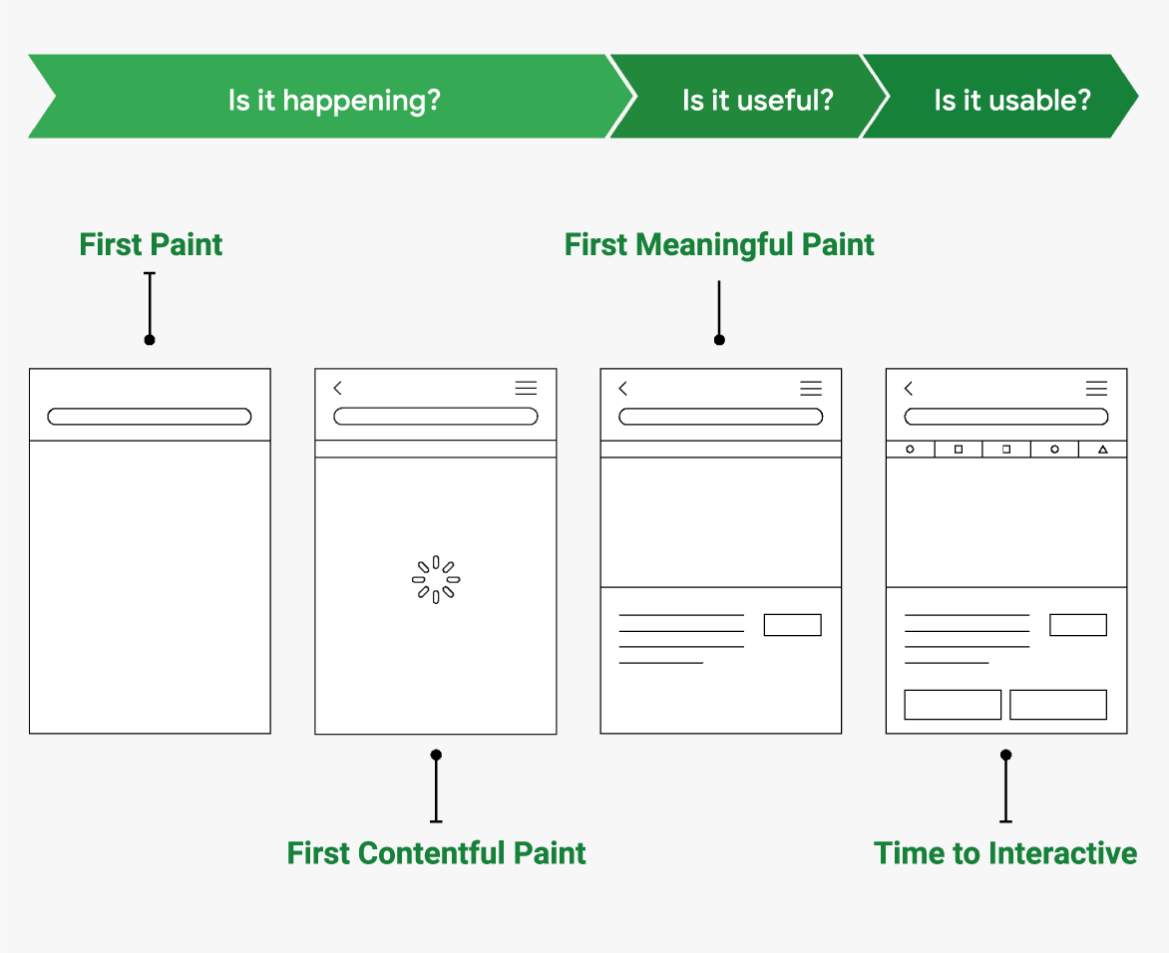
Obecně známým faktem je, že na rychlosti záleží. Důležité je ale porozumět tomu, co to vlastně znamená mít rychlou webovou stránku. Načítání není jeden okamžik v čase, ale zkušenost, kterou nelze zachytit žádnou jednotlivou metrikou. Během načítání existuje vícero okamžiků, které mohou ovlivnit uživatelský dojem, zda bude načítání vnímáno jako rychlé či pomalé. Pokud by byla věnována pozornost pouze na jeden moment, může dojít k přehlédnutí špatných výsledků momentů probíhajících ve zbylém čase. (Anstey, 2019)

Namísto měření zátěže pomocí pouze jedné metriky by měl být měřen každý okamžik v průběhu interakce s webem, který ovlivňuje uživatelovo vnímání rychlosti načítání. Když uživatel přijde na webovou stránku, obvykle hledá určité typy zpětné vazby. (Anstey, 2019)



Obrázek 1 Typy zpětné vazby (Anstey, 2019)

Tradiční metriky výkonu jako je doba načítání nebo doba „DOMContentLoaded“ jsou nespolehlivé, protože jejich výskyt může, ale také nemusí odpovídat těmto milníkům zpětné vazby. Objevily se proto další metriky, které by mohly být použity k pochopení toho, kdy stránka poskytuje uživatelům tuto zpětnou vazbu. (Anstey, 2019)



Obrázek 2 Metriky v průběhu načítání obsahu (Anstey, 2019)

Je důležité porozumět různým pohledům na problém, které tyto metriky nabízejí a poté určit ty, které jsou důležité pro uživatelský dojem v konkrétním případě. Některé firmy dokonce definují vlastní metriky zaměřené na specifické očekávání uživatelů jejich služeb. (Anstey, 2019)

I když je načítání více než jeden časový okamžik, stále může být užitečné mít jednu metriku pro účely zjednodušeného reportingu nebo srovnání. Index rychlosti a Lighthouse lze použít tímto způsobem. (Anstey, 2019)

3.9.2 Uživatelsky zaměřené výkonnostní metriky

Samotný výkon je relativní. Webové stránky se pro jednoho uživatele (s rychlým internetovým připojením a výkonným zařízením) mohou zdát rychlé, ale pro jiného uživatele (s pomalým internetovým připojením a méně výkonným zařízením) pomalé. Rozdíl lze pocítit i v případě, kdy čas kompletního načtení dvou stránek bude identický,

avšak jedna stránka načítá obsah progresivně a druhá stránka zobrazí obsah v momentě, až když je připraven celý. Dalším případem relativnosti výkonu je v případě, kdy dojde k rychlému načtení stránky, ale odezva na další interakci vyvolanou uživatelem reaguje pomalu nebo nereaguje vůbec. (Walton, 2019)

Čili pokud se mluví o výkonu, je důležité být přesný a odkazovat na výkon z hlediska objektivních kritérií, která lze kvantitativně měřit. Tato kritéria jsou nazývána metriky. Nicméně fakt, že je metrika založena na objektivních kritériích a lze ji kvantitativně měřit nutně neznamená, že jsou taková měření užitečná. (Walton, 2019)

3.9.2.1 Definice metrik

Historicky byl výkon webu měřen pomocí události „load“. Tato metoda přestala vyhovovat, a proto byl členy týmu Chrome ve spolupráci s W3C Web Performance Working Group vytvořen standardizovaný soubor nových API a metrik, které dokáží mnohem přesněji měřit, jak uživatelé pociťují výkon webové stránky. (Walton, 2019)

3.9.2.2 Měření metrik

Výkonnostní metriky jsou obecně měřeny dvěma způsoby. V laboratorních podmínkách, kdy je používáno různých nástrojů k simulaci načítání stránek v konzistentním, kontrolovaném prostředí. Laboratorní testování je zásadní při vývoji nových funkcí. Před vydáním funkcí do produkce není možné měřit jejich výkonnostní charakteristiky na skutečných uživatelích, takže testování v laboratoři je nejlepší způsob, jak zabránit výkonnostní regresi. (Walton, 2019)

Jelikož testování v laboratoři nemusí nutně odrážet, jak uživatelé vnímají web, používá se i druhý způsob, a to tzv. testování v praxi, kdy skuteční uživatelé navštěvují, načítají a provádějí interakce se stránkou. Výkon webu se může dramaticky lišit na možnostech zařízení uživatele a podmínkách připojení k síti. Odlišnosti mohou vznikat také z důvodu různorodé interakce uživatele se stránkou. Načítání stránek navíc nemusí být deterministické. Například weby načítající personalizovaný obsah nebo reklamy, mohou mít u jednotlivých uživatelů výrazně odlišné výkonnostní charakteristiky. Laboratorní testy tyto rozdíly nezachytí. Jediný způsob, jak zjistit skutečnou výkonnost webu pro uživatele je měření výkonu v momentě, kdy tito uživatelé načítají stránku a provádí s ní interakci.

Tento typ měření se běžně označuje jako monitorování skutečného uživatele. (Walton, 2019)

3.9.3 Typy metrik

Pro relevantnost měření, jak uživatelé vnímají výkon se používají tyto metriky:

- **Vnímaná rychlost načítání** – jak rychle se stránka může načíst a vykreslit na obrazovku všechny vizuální prvky.
- **Odezva načítání** – jak rychle se stránka může načíst a spustit jakýkoliv požadovaný JavaScript kód, aby komponenty rychle reagovaly na interakci uživatele.
- **Odezva za běhu** – jak rychle po načtení může stránka reagovat na interakci uživatele.
- **Vizuální stabilita** – posouvají se prvky na stránce způsobem, který uživatel neočekává a potenciálně narušují interakci.
- **Plynulost** – vykreslují se přechody a animace s konzistentní snímkovou frekvencí a plynule přecházejí z jednoho stavu do druhého.

Vzhledem ke všem výše uvedeným typům metrik výkonu je zřejmé, že žádná jednotlivá metrika nestačí k zachycení všech výkonnostních charakteristik stránky. (Walton, 2019)

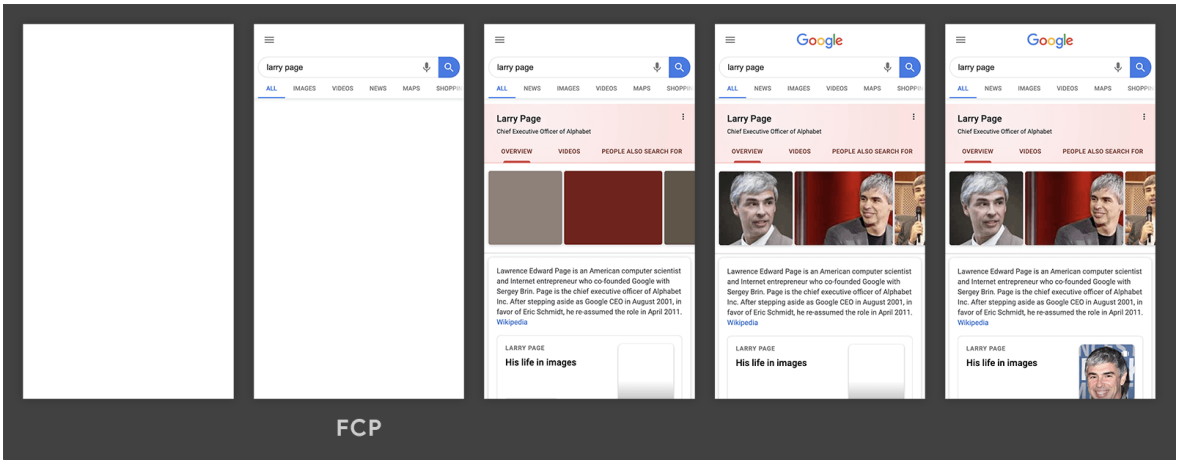
3.9.4 Metriky

V této kapitole bude věnována pozornost konkrétním metrikám.

3.9.4.1 First Contentful Paint (FCP)

First Contentful Paint (FCP) je metrika měřící čas od bodu, kdy se stránka začne načítat do okamžiku, kdy se jakákoliv část obsahu vykreslí na obrazovce. Pro tuto metriku se pod pojmem obsah rozumí text, obrázek (včetně obrázku na pozadí), <svg> a nebílé <canvas> elementy. (Walton, 2019)

Na obrázku níže zachycujícím osu načítání nastává FCP na druhém snímku, tehdy se na obrazovku vykreslují první textové a obrázkové prvky.



Obrázek 3 Zobrazení FCP během načítání obsahu (Walton, 2019)

FCP skóre

Pro zajištění dobrého uživatelského zážitku by se weby měly snažit o to, aby první vykreslení obsahu netrvalo déle než 1,8 sekundy. K získání jistoty, že je tohoto cíle u většiny uživatelů dosaženo je dobrým měřítkem 75. percentil načtení stránek, segmentovaný mezi mobilními a stolními zařízeními. (Walton, 2019)

Obrázek 4 FCP skóre (Walton, 2019)

3.9.4.2 Largest Contentful Paint (LCP)

Largest Contentful Paint (LCP) je metrika udávající dobu načítání největšího obrázku nebo textového bloku viditelného ve viewportu[3] od prvního momentu načítání stránky. Aktuální výčet typů elementů uvažovaných pro výpočet LCP je následující - <img>, <image> zabalený uvnitř <svg>, <video> (načtení náhledového obrázku), element s obrázkem na pozadí načítaný skrze funkci url() a block-level elementy obsahující textové

[3] Viewport je oblast aplikace, která se uživateli aktuálně zobrazuje na obrazovce

uzly nebo jiné potomky inline-level textových elementů. Omezení na tuto sadu prvků bylo záměrné ve snaze zachovat věci jednoduché. Další prvky jako plná podpora <svg> nebo plnohodnotné <video> mohou být přidány v budoucnu až proběhne další výzkum. (Pollard, 2023)

Kromě toho, že jsou brány v potaz jen některé prvky, je používána určitá heuristika k vyloučení určitých prvků, které budou pravděpodobně uživatelem považovány za bezpředmětné. V případě prohlížečů založených na Chromiu mezi ně patří:

- Elementy s neprůhledností 0, které jsou pro uživatele neviditelné.
- Elementy, které pokrývají celý viewport, které jsou považovány spíše jako pozadí než obsah.
- Zástupné obrázky nebo jiné obrázky s nízkou mírou entropie, které pravděpodobně neodrážejí skutečný obsah stránky.

(Pollard, 2023)

Webové prohlížeče budou s největší pravděpodobností pokračovat ve zdokonalování této heuristiky, aby zajistily, že dojde ke shodě s uživatelem v otázce, co je největší obsahový prvek. (Pollard, 2023)

Identifikace LCP elementu

Webové stránky jsou obvykle načítány v etapách a v důsledku toho je možné, že dojde ke změně největšího prvku na stránce. Aby prohlížeč byl schopen zvládnout tento potenciál změn, odešle PerformanceEntry typu LCP, který jakmile prohlížeč vykreslí první snímek identifikuje největší obsahový prvek. Po vykreslení dalších snímků odešle další PerformanceEntry kdykoliv se změní největší obsahový prvek. Je důležité poznamenat, že prvek lze považovat za největší až poté, co je vykreslen a viditelný pro uživatele. Pokud je aktuálně největší obsahový prvek odebrán z viewportu (nebo dokonce odstraněn z modelu DOM), i nadále zůstává největším prvkem, dokud nedojde k vykreslení většího prvku. Prohlížeč přestane hlásit nové záznamy, jakmile provede uživatel interakci se stránkou (kliknutím, posunutím nebo stiskem klávesy), protože interakce uživatele často mění, co je pro něho viditelné (zejména při posouvání). (Pollard, 2023)

Způsob určování velikosti prvku

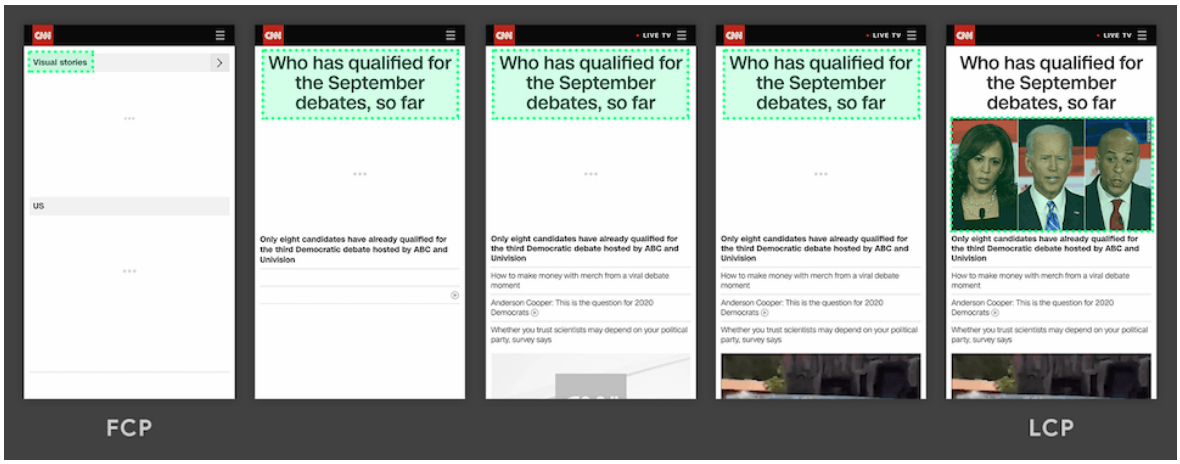
Velikost kontrolovaného prvku pro LCP je obvykle velikost, která je zobrazena uživateli ve viewportu. Pokud element přesahuje hranice viewportu nebo pokud je některý

z elementů ořízntutý nebo má neviditelné přetečení, tyto části se do velikosti nezapočítávají. V případě obrázků, jejichž velikost byla změněna oproti jejich skutečné velikosti je hlášena viditelná velikost nebo skutečná velikost podle toho, která z nich je menší. Například obrázky, které jsou zmenšeny do mnohem menší podoby, než jsou jejich skutečné rozměry budou hlásit velikost, ve které jsou zobrazeny. Zatímco obrázky, které jsou roztaženy do větších rozměrů budou hlásit pouze svou skutečnou velikost. U textových prvků se bere v úvahu pouze velikost jejich textových uzlů (nejmenší „obdélník“ zahrnující všechny textové uzly). U žádného z elementů se nebere v úvahu okraj, výplň nebo ohraničení aplikované prostřednictvím CSS. (Pollard, 2023)

Reakce na změnu rozložení a velikosti prvků

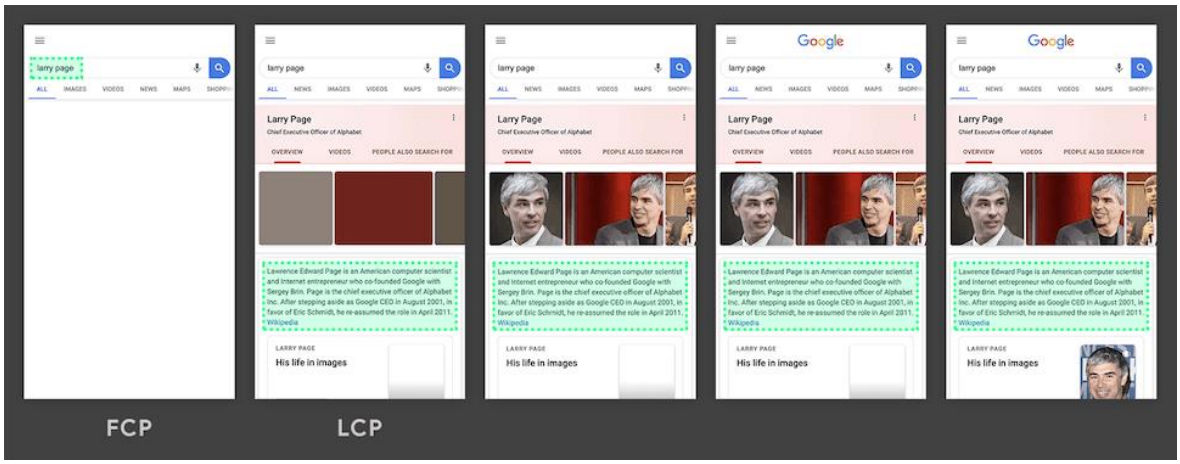
Kvůli udržení nízké režie výkonu při výpočtu a odesílání nových položek nejsou při změnách velikosti nebo pozice prvků generovány noví kandidáti LCP. Zohledněna je pouze počáteční velikost a poloha prvku ve viewportu. To znamená, že obrázky, které jsou zpočátku vykresleny mimo obrazovku a až posléze přejdou na obrazovku nemusí být hlášeny. Současně to znamená, že prvky původně vykreslené ve viewportu a následně přesunuty mimo zobrazovanou plochu budou stále hlásit svou původní velikost ve viewportu. (Pollard, 2023)

Na obrázku níže je s postupem načítání demonstrována změna největšího prvku.



Obrázek 5 Zobrazení LCP během načítání obsahu (Pollard, 2023)

Naopak na dalším obrázku je největší prvek vykreslen jako jeden z prvních, v dalším průběhu načítání se již nemění.



Obrázek 6 Zobrazení LCP během načítání obsahu (Pollard, 2023)

LCP skóre

Pro zajištění dobrého uživatelského zážitku by se weby měly snažit o to, aby vykreslení největšího obsahového prvku netrvalo déle než 1,8 sekundy. K získání jistoty, že je tohoto cíle u většiny uživatelů dosaženo je dobrým měřítkem 75. percentil načtení stránek, segmentovaný mezi mobilními a stolními zařízeními. (Pollard, 2023)

Obrázek 7 LCP skóre (Pollard, 2023)

3.9.4.3 First Input Delay (FID)

First Input Delay (FID) měří dobu od chvíle, kdy uživatel poprvé interaguje se stránkou (to znamená, když klikne na odkaz, klepne na tlačítko nebo použije vlastní ovládací prvek využívající JavaScript) do okamžiku, kdy je prohlížeč skutečně schopen začít zpracovávat manipulátory událostí v reakci na tuto interakci. (Walton, 2023)

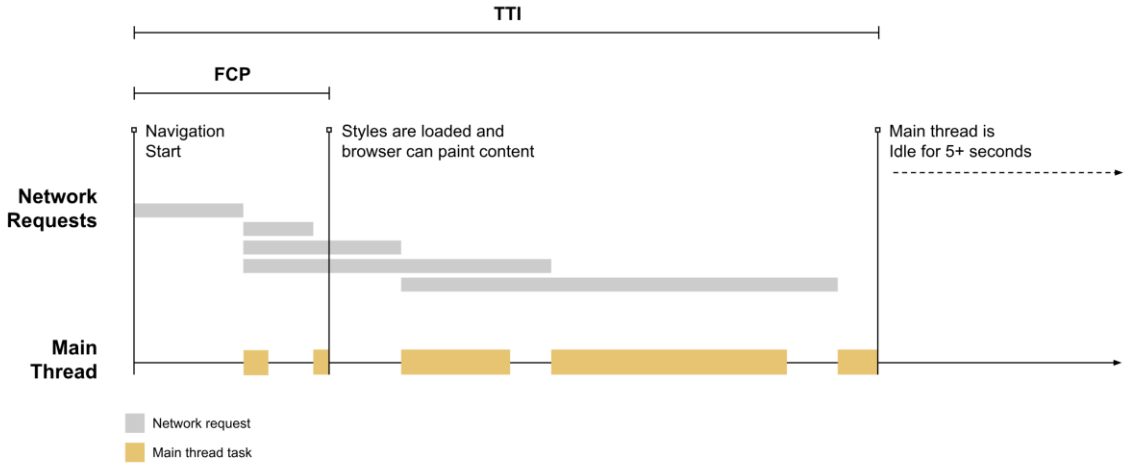
Obecně platí, že zpoždění vstupu (neboli latence vstupu) nastává, protože hlavní vlákno prohlížeče je zaneprázdněno něčím jiným, takže (zatím) nemůže uživateli odpovědět. Jedním z běžných důvodů, proč k tomu může docházet je, že je prohlížeč zaneprázdněn analýzou a spouštěním velkého JavaScript souboru načteného spouštěnou aplikací. Zatímco tuto úlohu zpracovává, nemůže spouštět žádné posluchače událostí

(event listeners), protože načítaný JavaScript může ovlivnit jeho další činnost. (Walton, 2023)

Je-li brána do úvahy časová osa typického načítání webové stránky, tak na další vizualizaci je zobrazena stránka zadávající několik síťových požadavků na zdroje (pravděpodobně CSS a JS soubory) a poté, co jsou tyto prostředky staženy jsou zpracovány v hlavním vláknu. To vyúsťuje do několika period, kde je hlavní vlákno zaneprázdněno. (Walton, 2023)

Obrázek 8 Časová osa s požadavky (Walton, 2023)

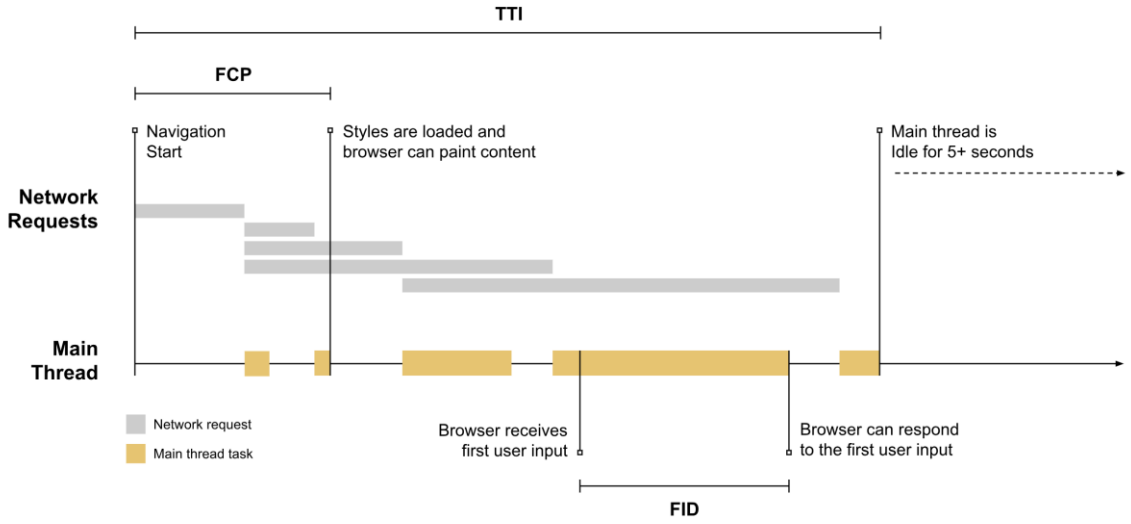
Dlouhé zpoždění prvního vstupu (FID) se obvykle vyskytuje mezi prvním vykreslením obsahu (FCP) a časem do interaktivity (TTI), protože stránka vykreslila část obsahu, ale ještě není spolehlivě interaktivní. Pro ilustraci byly na časovou osu přidány FCP a TTI.



Obrázek 9 FID na časové ose s požadavky (Walton, 2023)

Lze si povšimnout, že mezi FCP a TTI je poměrně dlouhá doba (včetně tří dlouhých úkolů). Pokud se uživatel během této doby pokusí interagovat se stránkou (například

kliknutím na odkaz), dojde ke zpoždění mezi okamžikem přijetí kliknutí a okamžikem, kdy je hlavní vlákno schopné odpovědět. V případě, kdy se uživatel pokusí o interakci na začátku nejdelšího úkolu, musí počkat, dokud není úkol plně zpracovaný, pak teprve může prohlížeč odpovědět na uživatelův požadavek. Doba, po kterou musí čekat je FID hodnotou pro konkrétního uživatele na konkrétní stránce. (Walton, 2023)



Obrázek 10 FID na časové ose s požadavky (Walton, 2023)

Pokud by však uživatel zadal svůj požadavek na zpracování během doby nečinnosti (v době mezi dvěma úkoly), prohlížeč by požadavek zpracoval okamžitě. Rozdíl ve vstupním zpoždění podtrhuje důležitost sledování distribuce FID hodnot při vytváření hlášení o metrice. (Walton, 2023)

Interakce bez Event listeneru

FID měří rozdíl mezi přijetím vstupní události a další nečinností hlavního vlákna. To znamená, že FID se měří i v případech, kdy event listener nebyl zaregistrován. Důvodem je to, že mnoho uživatelských interakcí nevyžaduje posluchače událostí, ale vyžaduje, aby bylo hlavní vlákno nečinné a možné spustit. Například všechny následující HTML prvky musí čekat na dokončení probíhajících úloh v hlavním vláknu, než budou reagovat na interakce uživatele.

- Textová pole, zaškrtávací políčka a přepínače (<input>, <textareas>)
- Výběr rozbalovací nabídky (<select>)
- Odkazy (<a>)

(Walton, 2023)

Měření pouze prvního vstupu

I když zpoždění od jakéhokoliv vstupu může vést ke špatné uživatelské zkušenosti, je doporučeno měřit zpoždění pouze prvního vstupu z několika důvodů:

- První zpoždění vstupu bude prvním dojmem uživatele z odezvy webu a první dojmy jsou rozhodující pro utváření celkového dojmu z kvality a spolehlivosti webu.
- Největší problémy s interaktivitou lze pozorovat při načítání stránky. Proto je považováno, že největší dopad na zlepšení celkového dojmu z interaktivity webu má vylepšení první interakce uživatele s webem.

(Walton, 2023)

Definice prvního vstupu

FID jako metrika měřící odezvu stránky se zaměřuje pouze na vstupní události z diskrétních akcí, jako kliknutí nebo stisk klávesy. Jiné interakce jako je posouvání a přibližování jsou kontinuální akce mající zcela jiná omezení výkonu (prohlížeče jsou často schopny skrýt svou latenci tím, že je spustí v samostatném vláknu). Jinými slovy se FID zaměřuje na responzivitu ve výkonnostním modelu RAIL, zatímco posouvání a přibližování se více vztahuje k animaci a jejich výkonnostní kvality by měly být hodnoceny samostatně. (Walton, 2023)

FID skóre

Pro zajištění dobrého uživatelského zážitku by se weby měly snažit o to, aby zpoždění prvního vstupu netrvalo déle než 100 milisekund. K získání jistoty, že je tohoto cíle u většiny uživatelů dosaženo je dobrým měřítkem 75. percentil načtení stránek, segmentovaný mezi mobilními a stolními zařízeními. (Walton, 2023)

Obrázek 11 FID skóre (Walton, 2023)

3.9.4.4 Interaction to Next Paint (INP)

Interaction to Next Paint (INP) je metrika posuzující celkovou odezvu stránky na interakce s uživatelem sledováním latence všech interakcí (kliknutí, klepnutí a stisky kláves), k nimž dochází po celou dobu trávení návštěvy uživatele na stránce. Konečná hodnota INP je nejdelší pozorovaná interakce (odlehlé hodnoty jsou ignorovány). (Wagner, 2023)

Jak již bylo zmíněno, INP je vypočítáváno na základě pozorování všech interakcí provedených se stránkou. U většiny webů je interakce s nejhorší latencí hlášena jako INP. U stránek s velkým počtem interakcí však může náhodné kolísání vést k neobvykle vysoké interakci na jinak responzivním webu. Čím víc interakcí, tím je pravděpodobnější, že taková situace nastane. K předcházení takových situací a tím zlepšení míry skutečné odezvy pro tyto typy stránek je používána metoda vyřazení nejvyšší hodnoty z každých 50 interakcí. 75. percentil všech zobrazení stránek pak udává hodnotu, s jakým nejhorším zážitkem se mohou uživatelé setkat. (Wagner, 2023)

Definice interakce

Interakce je skupina obslužných rutin událostí (event handlerů), které se spouštějí během stejného logického uživatelského gesta. Například interakce „klepnutí“ na zařízení s dotykovou obrazovkou zahrnují více událostí, jako je ukazatel nahoru, ukazatel dolů a kliknutí. Zde je potřeba dodat, že prostý pohyb myši a posouvání („scrollování“) v INP nehrají roli. Posouvání pomocí klávesnice (mezerník, page up, page down atd.) však zahrnuje stisky kláves, které mohou spouštět další události, které INP měří. Interakce může být řízena pomocí JavaScriptu, CSS, vestavěnými prvky prohlížeče (jako jsou prvky formuláře) nebo jejich kombinací. (Wagner, 2023)

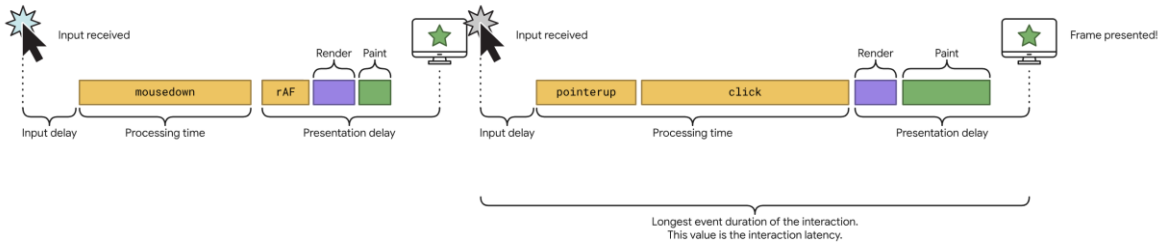
Latence interakce se skládá z jednoho nejdelšího trvání skupiny obslužných programů událostí, které řídí interakci, od okamžiku, kdy uživatel zahájí interakci, do okamžiku, kdy je další snímek prezentován s vizuální zpětnou vazbou. (Wagner, 2023)

Primárním hnacím motorem interaktivity je často JavaScript, i když prohlížeče poskytují interaktivitu prostřednictvím ovládacích prvků, které nejsou založeny na JavaScriptu, jako jsou zaškrtávací políčka, přepínače a ovládací prvky využívající CSS. (Wagner, 2023)

K interakcím dochází v hlavním dokumentu nebo v prvcích „iframe“ vložených do dokumentu – například kliknutím na přehrát ve vloženém videu. Koncoví uživatelé nepoznají, co je v iframe a co ne. K měření uživatelského dojmu na úvodní stránce je proto zapotřebí INP v rámci iframe prvků. (Wagner, 2023)

Interakce se mohou skládat ze dvou částí, z nichž každá obsahuje více událostí. Například stisk klávesy se skládá z událostí zmáčknutí klávesy, aktivace zmáčknutí klávesy, uvolnění klávesy. Interakce klepnutím obsahuje události „pointerup“ a „pointerdown“. Jako latence interakce je vybrána událost s nejdelším trváním v rámci interakce. (Wagner, 2023)

INP je vypočteno až v momentě, kdy uživatel opustil stránku. Výsledkem je jediná hodnota, která reprezentuje celkovou odezvu stránky během celého životního cyklu stránky. Nízké INP znamená, že stránka spolehlivě reaguje na vstup uživatele. (Wagner, 2023)



Obrázek 12 Grafické znázornění výpočtu INP (Wagner, 2023)

INP skóre

Označovat výsledky metriky měřící odezvu jako „dobré“ nebo „špatné“ je obtížné. Na jedné straně je snaha podporovat vývojové postupy upřednostňující odezvu, na druhou stranu je třeba počítat se skutečností, že existuje značná variabilita ve schopnostech zařízení, které lidé používají ke stanovení dosažitelných vývojových očekávání. (Wagner, 2023)

Aby bylo zajištěno, že je poskytován dobrý uživatelský dojem z odezvy, dobrým prahem k měření je 75. percentil načtení stránek zaznamenaných v produkci, segmentovaný mezi mobilními a stolními zařízeními:

- Stránka disponuje dobrou odezvou, pokud je hodnota INP menší nebo rovna 200 milisekund.

- Pokud je hodnota mezi 200 a 500 milisekundami, je třeba odezvu stránky vylepšit.
- U stránky s hodnotou vyšší než 500 milisekund hovoříme o špatné odezvě.

(Wagner, 2023)

Obrázek 13 INP skóre (Wagner, 2023)

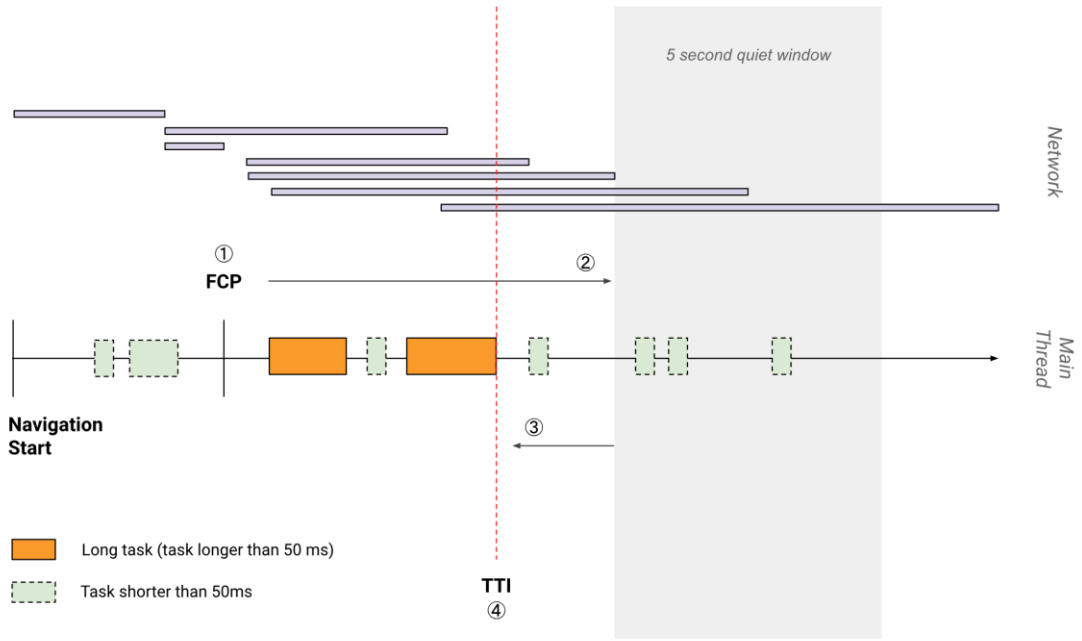
3.9.4.5 Time to Interactive (TTI)

Metrika Time to Interactive (TTI) měří čas od začátku načítání stránky do načtení jejích hlavních dílčích zdrojů a schopnosti spolehlivě a rychle reagovat na vstup uživatele. Pro výpočet TTI na základě sledování výkonu webové stránky je třeba postupovat následujícím způsobem:

1. Nejprve je třeba nalézt FCP.
2. Dále se posouvat v časové ose a hledat klidové okno v délce alespoň pěti sekund, kde klidové okno je definováno jako: neprobíhají žádné dlouhé úkoly a zároveň ne více než dva nedokončené požadavky GET ze sítě.
3. Dalším krokem je zpětné nalezení poslední dlouhé úlohy před klidovým oknem. Pokud nebyla nalezena žádná dlouhá úloha, je nahrazena FCP.
4. TTI je čas ukončení poslední dlouhé úlohy před klidovým oknem (nebo stejná hodnota jako FCP).

(Walton, 2023)

Na následujícím diagramu je vyobrazen průběh graficky:



Obrázek 14 Grafické znázornění TTI (Walton, 2023)

Historicky vývojáři optimalizovali stránky pro rychlé vykreslování, někdy na úkor TTI. Techniky jako je vykreslování na straně serveru (SSR) mohou vést ke scénářům, kdy stránka vypadá interaktivně (to znamená, že odkazy a tlačítka jsou na obrazovce viditelná), ale ve skutečnosti interaktivní není, protože je blokováno hlavní vlákno nebo protože JavaScript kód řídící tyto elementy nebyl načten. (Walton, 2023)

Když se uživatel pokusí interagovat se stránkou, která vypadá interaktivně, ale ve skutečnosti není, pravděpodobně zareaguje jedním ze dvou způsobů. V lepším případě bude uživatel pouze nespokojený s odezvou stránky, v horším případě může nabýt dojmu, že je stránka poškozená a opustit ji. Druhému scénáři se lze vyhnout buď minimalizováním rozdílu mezi FCP a TTI, případně alespoň umístit na stránky vizuální indikátory, které dají najevo, že komponenty na stránce zatím nejsou interaktivní. (Walton, 2023)

TTI skóre

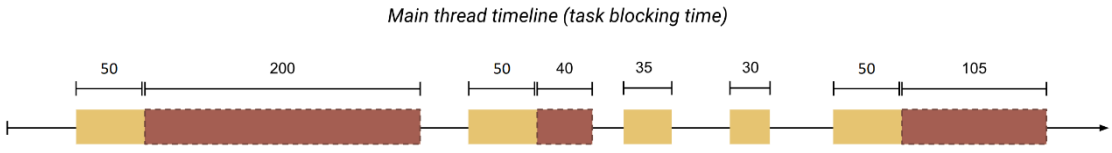
Pro zajištění dobrého uživatelského dojmu by se webové stránky měly při testování na průměrném mobilním hardwaru snažit dosáhnout TTI do 5 sekund. (Walton, 2023)

Obrázek 15 TTI skóre (Walton, 2023)

3.9.4.6 Total Blocking Time (TBT)

Metrika Total Blocking Time (TBT) měří celkovou dobu mezi FCP a TTI, kdy bylo hlavní vlákno zablokováno na dostatečně dlouhou dobu, aby zapříčinilo narušení responzivnosti vstupu. Hlavní vlákno je považováno za „zablokované“ vždy, když existuje dlouhá úloha – úloha běžící v hlavním vláknu déle než 50 milisekund. Vlákno je označováno jako „zablokované“, protože prohlížeč nemůže přerušit probíhající úlohu. V případě, že uživatel interaguje se stránkou uprostřed dlouhé úlohy, prohlížeč musí počkat na její dokončení, než bude moci reagovat. (Walton, 2023)

Pokud je úloha dostatečně dlouhá (cokoliv nad 50 milisekund), je pravděpodobné, že si uživatel zpoždění všimne a bude stránku považovat za pomalou a zastaralou. Doba blokování dané dlouhé úlohy je její trvání přesahující 50 milisekund. Celková doba blokování stránky je pak součet doby blokování pro každou dlouhou úlohu, ke které dojde mezi FCP a TTI. Na následující časové ose je vyobrazeno 5 úkolu, přičemž 3 z nich jsou dlouhé (přesahují dobu 50 milisekund). Dlouhé úkoly jsou graficky rozděleny na dvě části, kde druhá část vystihuje dobu blokace. (Walton, 2023)



Obrázek 16 Grafické znázornění blokace hlavního vlákna (Walton, 2023)

Celkový čas strávený během úloh v hlavním vlákně je 560 ms, pouze 345 ms z tohoto času je považováno za dobu blokování.

	Doba trvání úkolu [ms]	Doba blokace vlákna [ms]
1. Úkol	250	200
2. Úkol	90	40
3. Úkol	35	0
4. Úkol	30	0
5. Úkol	155	105
Celkový blokovaný čas		345

Tabulka 1 Doba blokace hlavního vlákna (Walton, 2023)

Vztah mezi TBT a TTI

TBT je skvělá doplňková metrika pro TTI, protože pomáhá kvantifikovat závažnost neinteraktivní stránky, než se stane spolehlivě interaktivní. TTI považuje stránku za spolehlivě interaktivní, pokud hlavní vlákno neobsahuje dlouhé úkoly po dobu alespoň pěti sekund. To znamená, že tři 51 ms úlohy rozložené do 10 sekund mohou posunout TTI stejně daleko, jako jediná 10 sekund dlouhá úloha. Tyto dva scénáře by se ale uživateli snažícímu se interagovat se stránkou zdály velmi odlišné. V prvním případě by tři úlohy 51 ms měly TBT 3 ms. Zatímco jeden 10 sekund dlouhý úkol by měl TBT 9950 ms. Větší hodnota TBT ve druhém případě kvantifikuje horší zkušenost. (Walton, 2023)

TBT skóre

Pro zajištění dobrého uživatelského dojmu by se webové stránky měly při testování na průměrném mobilním hardwaru snažit dosahovat celkové doby blokování menší než 200 milisekund. (Walton, 2023)

Obrázek 17 TBT skóre (Walton, 2023)

3.9.4.7 Cumulative Layout Shift (CLS)

Cumulative Layout Shift (CLS) je měřítkem největšího nárůstu změny skóre posunu rozložení stránky pro každý neočekávaný posun rozložení, ke kterému dojde

během celé životnosti stránky. K posunu rozložení dojde vždy, když viditelný prvek změní svou polohu z jednoho vykresleného snímku na další. Shluk posunů rozložení, známý jako okno relace nastane tehdy, když dojde k jednomu nebo více jednotlivým posunům rozložení v rychlém sledu s méně než 1 sekundou mezi každým posunem a maximálně 5 sekundami po celou dobu trvání okna. Největší dávka je okno relace s maximálním kumulativním skóre všech posunů rozvržení v tomto okně. (Mihajlija, 2023)

Posun rozložení

Posuny rozložení jsou definovány rozhraním „Layout Instability API“, které hlásí položky změny rozložení, kdykoliv prvek viditelný ve viewportu změní svou počáteční pozici mezi dvěma snímky. Takové prvky jsou označovány za nestálé. K posunům rozložení dochází pouze tehdy, změní-li stávající prvky svou polohu. Pokud je do DOM přidán nový prvek nebo stávající prvek změní velikost, nepočítá se to jako posun rozložení – dokud změna nezpůsobí změnu počáteční pozice ostatních viditelných prvků. (Mihajlija, 2023)

Výpočet skóre posunu rozložení

Při výpočtu skóre posunu rozložení (layout shift score) prohlížeč sleduje velikost viewportu a v něm pohyb nestálých prvků mezi dvěma vykreslenými snímky. Skóre posunu rozložení je součinem dvou měření tohoto pohybu – zlomku dopadu (impact fraction) a zlomku vzdálenosti (distance fraction). Vzorec pro výpočet je následující:

- *layout shift score = impact fraction × distance fraction*

(Mihajlija, 2023)

Impact fraction měří, jak nestálé prvky ovlivňují oblast výřezu obrazovky mezi dvěma snímky. Sjednocení viditelných oblastí všech nestálých prvků z předchozího a aktuálního snímku jako zlomek celkové plochy výřezu je zlomkem dopadu pro aktuální snímek. Například bude-li prvek zabírat polovinu viewportu v jednom snímku a poté se v dalším snímku prvek posune o 25 % výšky výřezu dolů, spojením viditelných oblastí prvku z obou snímků představuje výslednou hodnotu 75 % celkového výřezu. Podíl dopadu bude 0,75. (Mihajlija, 2023)

Druhá část rovnice skóre posunu rozložení měří vzdálenost, o kterou se přesunuly nestálé prvky relativně k výřezu. Distance fraction je největší vzdálenost, o kterou se jakýkoliv nestálý prvek posunul (ať už horizontálně nebo vertikálně) dělena největším

rozměrem výřezu (šířka nebo výška podle toho, která hodnota je větší). Při použití stejného příkladu, kdy se nestálý prvek posunul o 25 % výšky výřezu a největším rozměrem výřezu je výška, zlomek vzdálenosti činí 0,25. V tomto příkladu je tedy Impact fraction 0,75 a Distance fraction 0,25, takže výsledné skóre posunu rozložení činí *0,75 × 0,25 = 0,1875*. (Mihajlija, 2023)

CSL skóre

Pro zajištění dobrého uživatelského zážitku by se weby měly snažit o dosažení CLS skóre 0,1 nebo méně. K získání jistoty, že je tohoto cíle u většiny uživatelů dosaženo je dobrým měřítkem 75. percentil načtení stránek, segmentovaný mezi mobilními a stolními zařízeními. (Mihajlija, 2023)

Obrázek 18 CLS skóre (Mihajlija, 2023)

3.9.4.8 Time to First Byte (TTFB)

Time to First Byte (TTFB) je metrika měřící čas mezi požadavkem na zdroj a okamžikem, kdy začne přicházet první „byte“ odpovědi. TFFB je součet následujících fází požadavku:

- Čas přesměrování
- Čas spuštění tzv. „Service worker[4]“ (pokud je to možné)
- DNS lookup
- Připojení a vyjednávání TLS
- Požadavek až do okamžiku, kdy dorazil první byte odpovědi

[4] Service worker je specializované JavaScript aktivum fungující jako proxy mezi webovými prohlížeči a webovými servery. Jejich cílem je zlepšit spolehlivost poskytování offline přístupu a zvýšit výkon stránky.

- Snížení latence v době nastavování připojení a na back-endu přispěje k nižšímu TTFB

(Wagner, 2021)

TTFB skóre

Protože TTFB předchází metrikám zaměřeným na uživatele jako je FCP a LCP, doporučuje se, aby server reagoval na požadavky navigace dostatečně rychle, aby 75. percentil uživatelů zaznamenal FCP v rozmezí „dobré“ hodnoty. Webové stránky by se měly snažit dosáhnout doby do prvního bytu 0,8 sekundy nebo méně. (Wagner, 2021)

Obrázek 19 TTFB skóre (Wagner, 2021)

3.9.4.9 Speed Index (SI)

Speed Index (SI) je metrika výkonu načítání stránky, která měří, jak rychle se obsah stránky viditelně zaplní. Index rychlosti závisí na velikosti zobrazované oblasti (viewport) a vyjadřuje se v milisekundách, kdy je kratší doba ohodnocena lepším skóre. (Speed Index, 2024)

Speed Index je vypočítáván podle toho, jaké procento stránky je vizuálně dokončeno každých 100 ms, dokud není stránka vizuálně dokončena. Celkové skóre je součtem jednotlivých deseti intervalů během sekundy z procenta obrazovky, která není vizuálně zcela úplná. (Speed Index, 2024)

SI skóre

Pro zajištění dobrého uživatelského dojmu by webové stránky měly dosahovat indexu rychlosti méně než 3,4 sekundy. (Core Web Vitals — A Google Initiative to measure performance, 2024)

Obrázek 20 Speed Index (Core Web Vitals — A Google Initiative to measure performance, 2024)

3.10 Optimalizace CSS a JS

Pro rychlé načítání webových stránek je klíčová optimalizace. K té je třeba přistupovat komplexně, tedy od aplikační vrstvy (back-end) až po prezentační vrstvu (front-end). V rámci optimalizace front-endu je dobré se zaměřit na následující body, které mohou značně snížit dobu načítání. (Bin Uzayr, 2022)

3.10.1 Optimalizace CSS

Optimalizace obrázků: Obrázky svou velikostí, v závislosti na kvalitě, dosahují jednotek megabytů, což je pro rychlost načítání velký problém. Jedním řešením je ručně snížit kvalitu obrázku, zvláště pokud se na stránkách zobrazuje jen jako miniatura. Ošetřením na úrovni CSS kódu lze využít vlastnosti loading=lazy, což zamezí načítání obrázků, které nejsou ve viewportu (obrázkům ve vrchní části stránky se nikdy nenastavuje). Dále není doporučováno dávat obrázek jako pozadí stránky (opět kvůli načítání většího objemu dat a problémům s přizpůsobením obrázku na různé velikosti zařízení) a pokud je to možné, upřednostnit SVG grafiku. (Sewell, 2022)

Odebrání nepoužívaného kódu: Při úpravách webových stránek často dochází k ponechávání již nevyužívaných stylů stále v CSS dokumentech. Ačkoli může být hledání takových stylů složité, vyplatí se ohlídat a zabránit narůstání velikosti dokumentu. (Sewell, 2022)

Zřetězení a minifikace: Zřetězením dojde ke spojení více CSS souborů do jednoho. Tím dojde ke snížení počtu dotazů na server, a tak dojde k rychlejšímu načtení a menšímu zatížení serveru. Následně je možné se souborem provést minifikaci a kompresi, což znamená, že dojde ke zkrácení kódu odebráním veškerých zbytečných znaků, které prohlížeč pro správné zobrazení nepotřebuje (mezery, zalamování textu,

komentáře apod.). Důležité je zachovat správnou syntaxi. Takto lze zkrátit dokument v průměru o 20 %. (Sewell, 2022)

Prioritizace: Důležité části CSS by měly být načítány jako první, aby se uživateli co nejdříve vykreslily klíčové prvky stránky i přesto, že dojde k přetížení serveru a poklesu jeho výkonnosti. Prioritizaci lze provádět více způsoby, například vložením klíčové části CSS přímo do hlavičky stránky, vyvarováním se používání pravidla @import. To blokuje vykreslování a snižuje rychlost webové stránky tím, že se každý importovaný externí soubor načítá samostatně, místo aby se načítal paralelně se všemi ostatními soubory potřebnými k vykreslení konkrétní stránky. Efektivnějším způsobem je použití více značek <link>, jelikož jsou soubory načítány paralelně. (Hazeez, 2022) (Sewell, 2022)

Optimalizace fontů: Načítání fontů má přímý vliv na FCP, případně LCP. Jelikož se do momentu načtení fontů nezobrazuje text, může při jeho zpožděném načtení výrazně prodloužit dobu do vykreslení obsahu a zhoršit tak uživatelský zážitek. Pomocí vlastnosti font-display lze nastavit, jak bude stránka postupovat v případě, kdy nedojde k načtení fontů. (Bin Uzayr, 2022)

3.10.2 Optimalizace JavaScriptu

Minifikace a sdružování: Minifikace a sdružování jsou základní techniky ke snížení velikosti JavaScript souborů a minimalizaci počtu HTTP požadavků. Minifikací dochází k odstranění nepotřebných znaků, jakými jsou mezery a komentáře, a sdružování spojuje více souborů do jednoho. (Prajapati, 2023)

Odstranění přebytečného kódu: Kód by měl být sám o sobě napsaný efektivně, aby byla jeho velikost co nejmenší. Zároveň je třeba dávat pozor, aby byly odebrány skripty, které již nejsou pro webové stránky relevantní. (Prajapati, 2023)

Podmíněné a líné načítání: Takzvané „líné načítání“ (z anglického „lazy loading“) je technika, při které se JavaScript soubory načítají pouze v případě, že jsou potřeba. Tedy pokud je vyvolána specifická akce nebo událost na stránce. Podmíněné načítání pak dovoluje načítat JavaScript soubory selektivně v závislosti na specifické podmínce. Například načítání různých skriptů v závislosti na typu koncového zařízení, schopnostech prohlížeče nebo uživatelské interakce. Tím, že se při prvotním načtení stránky načítají pouze kritické skripty se snižuje počáteční doba načítání. (Ene Anyebe, 2023)

Asynchronní a odložené načítání: Javascript soubory se ve výchozím nastavení načítají synchronně, což znamená, že blokují vykreslování webové stránky, dokud se skript plně nenačte a nespustí. Techniky asynchronního a odloženého načítání umožňují načítání souborů JavaScriptu nezávisle na procesu vykreslování stránky, čímž se minimalizuje dopad na dobu načítání. Asynchronní načítání zajišťuje, že skript je načten a spuštěn, jakmile je k dispozici, zatímco odložené načítání zdržuje provedení, dokud není dokončena analýza HTML. (Ene Anyebe, 2023)

Optimalizace kódu: K optimalizaci kódu pro zlepšení výkonu lze využít různých technik v závislosti na aktuálních potřebách. (Prajapati, 2023)

3.11 Přehled obdobných řešení a doporučení

Srovnání jednotlivých CSS frameworků bylo provedeno již mnohokrát. Obvykle jsou testy prováděny jednou ročně a zabývají se srovnáním nejvíce populárních frameworků, nicméně studie zabývající se přímo tématem měření doby načtení webového obsahu se neprovádí.

3.11.1 Výsledky zveřejněných testů CSS frameworků

V této kapitole budou prezentovány výsledky testů provedené v uplynulém roce. Následující srovnání bude zaměřeno pouze na nejpopulárnější frameworky s obecným zaměřením.

Obrázek 21 Rozdělení CSS frameworků (Semah, 2023)

3.11.1.1Bootstrap

Jako jedna z nejlepších CSS knihoven Bootstrap nabízí konzistentní a otestovanou kódovou základnu, na kterou se lze spolehnout pro dosažení konzistentních výsledků. Kompromisem je, že ponechává pouze malý prostor pro flexibilitu návrhu.

Velkým benefitem je rozsah poskytovaných tříd a komponent. Bootstrap u většiny běžných návrhů nevyžaduje dodatečné rozšiřování stylů, díky čemuž si udržuje vysokou míru konzistence. Toto napomáhá i lepší spolupráci při vývoji.

Díky své komplexnosti nalezne své uplatnění v širokém spektru projektů, především při tvorbě:

- Webové stránky a blogy
- Firemní webové stránky
- E-commerce platformy
- Administrativní rozhraní
- Mobilní aplikace
- Stránky zaměřené na obsah

Klady

- Vestavěná podpora responzivního designu
- Rozsáhlá knihovna komponent pro stavbu webu
- Snadno přizpůsobitelné komponenty
- Podpora SASS i LESS
- Dobrá dokumentace
- Velká podpora komunity

Zápory

- Omezená flexibilita návrhů díky před-designovaným komponentám
- Velká velikost souborů zvyšující dobu stahování

(Adarsh, 2023)

3.11.1.2 Foundation

Foundation poskytuje vývojářům vyšší míru přizpůsobitelnosti. Díky tomu je možné dosáhnout rozmanitějších vzhledů stránek, ovšem za cenu delšího času potřebného k seznámení se s používáním frameworku. V případě přidání interaktivity na stránky lze Foundation rozšířit o Javascript pluginy.

Obecně platí, že Foundation je vhodným nástrojem pro vývoj moderních, responzivních a uživatelsky přívětivých webových stránek a aplikací. Konkrétními příklady jsou:

- Webové stránky a aplikace
- Mobilní aplikace
- E-commerce platformy
- Administrativní rozhraní
- Stránky zaměřené na obsah
- Webové aplikace s vysokým výkonem

Klady

- Vysoká míra flexibility. Široká škála modulárních a flexibilních komponent má minimální stylování a lze je snadno přizpůsobit
- Obsahuje pokročilý systém pro responzivitu obrázků
- Masivní sada nástrojů, kterých je nabízeno více, než CSS framework vyžaduje. Jeho rozsáhlá modulární sbírka nástrojů pomáhá řešit problémy většiny front-end vývojářů. Nabízí například samostatné prvky pro webové stránky a e-maily
- Poskytuje rozhraní příkazového řádku (CLI), které je užitečné při práci se svazky modulů
- Nabídka tréninkových programů a konzultací

Zápory

- Nadměrné užívání JavaScriptu k dosahování určitých stylů a efektů, což může vést k potenciálnímu zpomalení načítání stránek
- Příliš komplexní, pro nové uživatele může být složité

(Adarsh, 2023)

3.11.1.3Bulma

Bulma je framework vytvořený jako minimalistická alternativa dalších CSS frameworků. Bulma neobsahuje žádné JavaScript komponenty. Je tak velmi lehký, rychlý a optimalizovaný pro výkon.

Přestože je Bulma označována za lehký framework, stále nabízí velké množství komponent pro vývoj webových stránek a řadí se tak mezi frameworky pro všeobecné použití. Využít ji lze díky své rychlosti, modulárnímu a responzivnímu designu například pro vývoj:

- Moderní webové stránky
- Webové aplikace
- Landing page
- Blogy a portfolia
- Projekty s vlastním designem
- Projekty vyžadující lehký kód

Klady

- Extrémně čitelné třídy díky jejich intuitivnímu pojmenování
- Jednoduchý, přesto moderní design
- Mezi univerzálními frameworky se řadí mezi „Lightweight“ neboli odlehčené frameworky, a i díky tomu je velice rychlý
- Neobsahuje funkce založené na JavaScriptu. Díky tomu může být integrován s JavaScript frameworky, jako například Vue nebo React
- Jednoduché přizpůsobení vlastností pomocí SASS

Zápory

- Menší podpora komunity
- Komplexita není takové úrovni, některým vývojářům mohou scházet určité funkce, především v oblasti přístupnosti a na podnikové úrovni

(Semah, 2023)

3.11.1.4Fomantic UI

Fomantic UI je navržen tak, aby byl rychlý, efektivní a poskytoval podporu přístupnosti. JavaScript funkce lze upravovat pomocí předdefinovaných parametrů a tím si snadno a rychle přizpůsobit chování interaktivních prvků. Fomantic UI je vhodným nástrojem pro vývoj například následujících projektů:

- Webové stránky a aplikace

- Administrativní rozhraní
- Mobilní aplikace
- Projekty s vlastním designem
- Projekty vyžadující zachování kompatibility

Klady

- Zaměření na přístupnost
- Intuitivní dědičnost a vysoká míra tematické proměnné podporující svobodu návrhu
- Jednoduché použití
- Mobile-first přístup

Zápory

- Menší míra podpory komunity
- Horší dokumentace
- Omezená podpora JavaScript funkcí

(Harnessing The Power Of Top 15 CSS Frameworks: Responsive Web Design Unleashed, 2023)

3.11.1.5 Blaze UI

Blaze UI se zaměřuje na škálovatelnost a udržovatelnost projektů. Přestože je Blaze UI méně populární framework a poskytuje méně zdrojů, obsahuje veškeré styly potřebné k vytvoření elegantních a responzivních webových rozhraní. Použití Blaze UI je vhodnou volbou například u následujících projektů:

- Webová aplikace
- Administrativní rozhraní
- Webové stránky
- E-commerce platformy
- Stránky zaměřené na obsah
- Projekty s vlastním designem

Klady

- Odlehčený a rychlý
- Zaměření na přístupnost
- Jednoduchá úprava prvků

Zápory

- Omezená podpora komunity a méně zdrojů
- Omezená podpora JavaScript funkcí

(Adarsh, 2023)

3.11.2 Vliv CSS na dobu načítání

Rozdíl mezi optimalizovaným a neoptimalizovaným CSS lze pozorovat na době načítání stránky. Ačkoliv na stránky působí větší množství různých vlivů, které mají dopad na dobu načítání, CSS patří mezi významnější. Jelikož je většina CSS stylů tzv. „render-blocking“, což znamená, že se uživateli nezobrazí žádný obsah, jelikož se čeká, než budou styly načteny. Do té doby je zobrazení obsahu blokováno. V takovém případě je velmi důležité, aby byla načtena alespoň část obsahu a bylo zřejmé, že se stránka načítá, jelikož uživatelé obvykle stránky bez náznaku aktivity opouští během prvních pár vteřin. Míra, kterou jsou stránky ovlivněny v závislosti na kvalitě zpracování CSS se liší v závislosti na komplexnosti stránek, a především na míře optimalizace. Nelze tedy obecně stanovit rozdíl vyjádřený časovou jednotkou, ale je třeba posuzovat každou situaci individuálně.

4 Vlastní práce

Kapitola vlastní práce se zabývá celým procesem srovnání. Celý proces začíná návrhem experimentálních stránek zaměřených na obsažení rozmanitosti prvků. Dalším krokem je výběr CSS frameworků, které budou podrobeny zkoumání a vytvoření experimentálních stránek. Další důležitou částí je návrh kritérií, které budou sledovány, a jejich vah a na základě jejich výsledků budou stanoveny závěry. Pro sledování kritérií je zapotřebí vybrat vhodné měřící prostředky a pokud je třeba, tak provést potřebné modifikace.

4.1 Návrh experimentálních stránek

K otestování jednotlivých frameworků podle zvolených metrik bylo zapotřebí vytvořit se všemi zvolenými frameworky experimentální webové stránky. Z tohoto důvodu bylo zapotřebí předem vytvořit návrh zmíněných experimentálních webových stránek.

Při návrhu byl kladen důraz na implementaci různorodých prvků, aby bylo možné sledovat možnost různorodého přístupu k vytváření takových prvků. Současně byla snaha použít pouze takové prvky, které jsou obsaženy ve všech zvolených frameworcích.

Jelikož se jedná pouze o stránky určené k testování, tak neobsahují copywriting. Použitá textace a obrázky slouží pouze jako výplň. Na výsledek testování nemá tato skutečnost žádný vliv.

4.1.1 Výběr typu stránky

Před samotnou tvorbou návrhu bylo třeba zvolit o jaký typ stránky se bude jednat. Při rozhodování byla zásadním argumentem otázka, „V jaké oblasti má špatná doba načítání nejvíce negativní dopady?“ Z tohoto hlediska byla upřena pozornost na stránky e-commerce. V tomto případě může pomalé načítání výrazně ovlivnit úspěšnost prodeje a tím i úspěšnost celé společnosti. Se vzrůstající dobou načítání klesá kvalita uživatelského zážitku, společně s tím klesá důvěra v obchod a tím dochází k poklesu konverzního poměru. Až 60 % návštěvníků stránky opouští, pokud jejich prvotní načtení trvá déle, než 5 sekund. Samotnou návštěvnost ovlivňuje i fakt, že webové vyhledávače upřednostňují stránky s vyšší mírou výkonnosti.

V rámci návrhu experimentálních webových stránek byl vytvořen web obsahující 3 stránky. Mimo úvodní stránku je dále obsažena stránka s detailem produktu a stránka s popisem společnosti.

Rozložení všech níže uvedených stránek a jejich prvků je navrženo pro zařízení disponující většími rozměry obrazovky, konkrétně tedy obrazovky notebooků a stolních počítačů. Důvodem je větší míra možné rozmanitosti rozložení prvků na stránce na obrazovkách větších rozměrů, nežli je tomu například u obrazovek mobilních zařízení. Avšak v některých případech je explicitně specifikováno rozložení i pro mobilní zařízení.

4.1.2 Úvodní stránka

Úvodní stránka má v tomto případě několik funkcí. Zejména má za úkol zaujmout potenciální zákazníky, představit nabízený sortiment a zajistit určitou míru propagace. Dále slouží jako rozcestník pro další stránky webu.

Na stránce se nachází vícero různorodých prvků a celou strukturu lze rozdělit na 3 části, a to hlavičku, hlavní část a patičku, přičemž hlavní část je složena z více menších sekcí. Popis jednotlivých sekcí a prvků je uveden níže.

4.1.2.1 Hlavička

Hlavička se skládá primárně ze dvou částí. Z loga společnosti a navigace. Logo společnosti funguje současně jako proklik na úvodní stránku. Navigace poté zajišťuje možnost přesměrování na další stránky a vyhledávání na webu. Prvky navigace jsou horizontálně orientovány. Celá hlavička je po celou dobu přichycena na horním okraji stránky, aby byla dosažitelná z kterékoliv části stránky.

Obrázek 22 Hlavička

Pro mobilní zobrazení se pak navigace schová a bude se zobrazovat jako tzv. „hamburger-menu“. Výjimkou je ikona nákupního košíku, která zůstává viditelná bez nutnosti rozbalit menu a nachází se vlevo od ikony menu. Rozbalené menu vychází z horní části obrazovky a obsahuje vertikálně řazené položky navigace.

Obrázek 23 Rozbalené mobilní menu

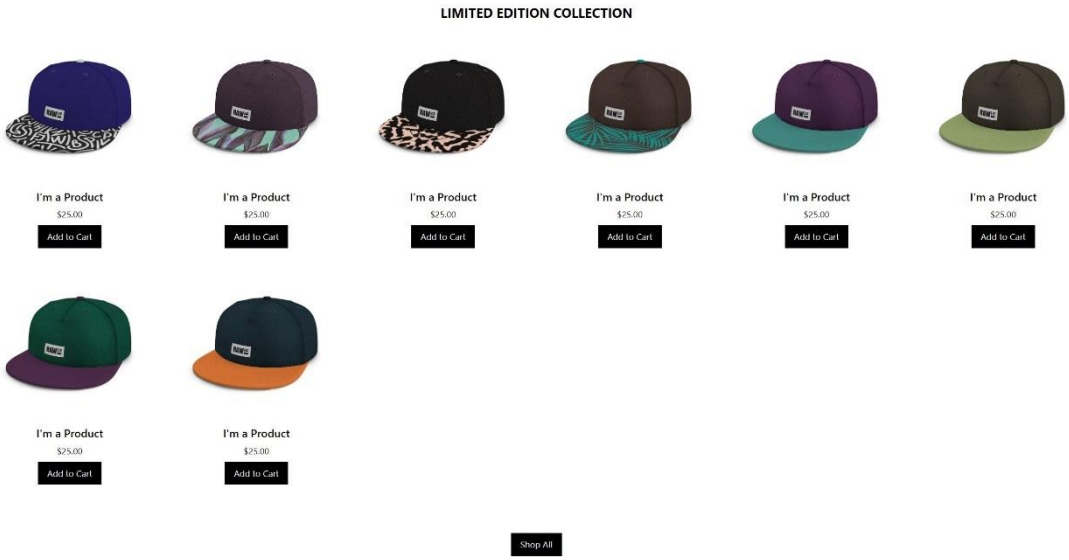
4.1.2.2 Hlavní část

Hlavní část obsahuje další sekce s obsahem. Těmito sekcemi jsou: hero sekce, sekce s produkty a sekce s prezentováním produktů skrze fotografie.

Hero sekce – Hero sekce je dominantním prvkem na stránce a plní svoji funkci upoutat pozornost uživatele. Součástí hero sekce je CTA tlačítko vyzývající uživatele k navštívení stránky s produkty. Při namíření kurzoru na tlačítko dochází ke změně jeho barevného zobrazení.

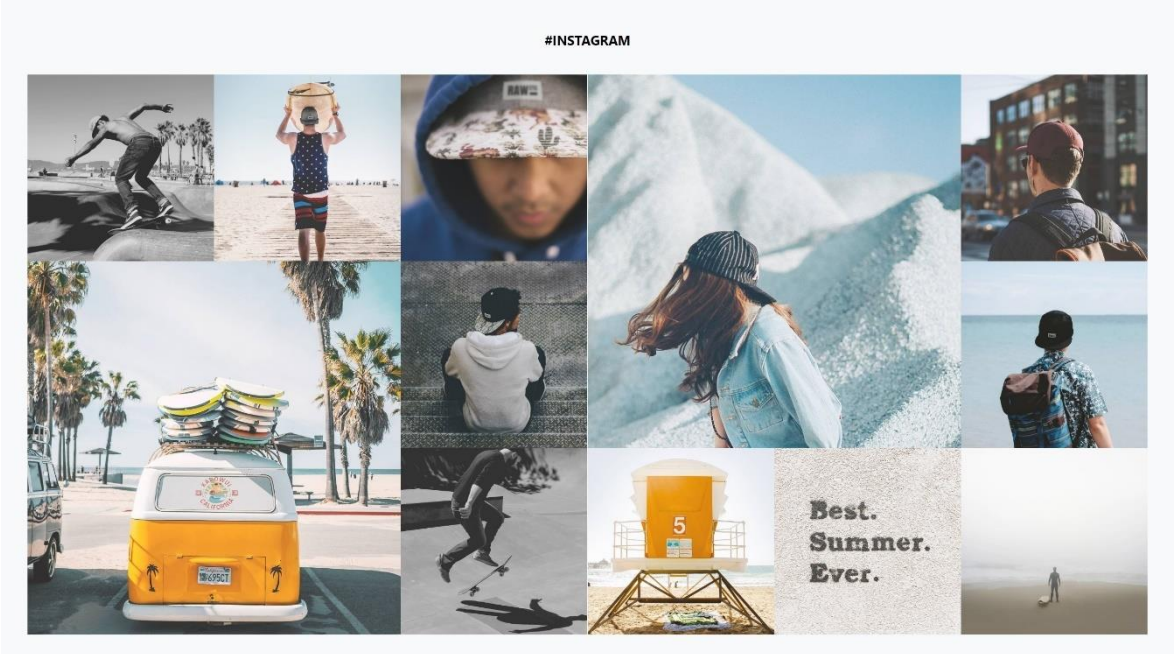
Obrázek 24 Hero sekce

Sekce s produkty – V této sekci se nachází výběr produktů. Každý produkt disponuje produktovou fotografií, názvem produktu, cenou produktu a CTA tlačítkem pro vložení produktu do košíku. Skrze produktovou fotografii a název produktu je realizováno přesměrování na stránku s detailem produktu. Ve spodní části této sekce se nachází další CTA tlačítko pro přesměrování na seznam všech produktů. Všechna tlačítka v této sekci invertují své barevné zobrazení v případě, jsou-li zacílena kurzorem.



Obrázek 25 Sekce s produkty

Sekce pro prezentaci produktů – Tato sekce obsahuje mřížkové uspořádání fotografií prezentujících prodávané produkty.



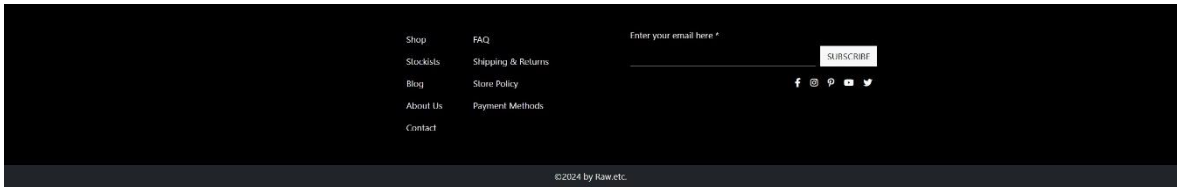
Obrázek 26 Sekce pro prezentaci produktů

Tyto fotografie mají funkci tlačítek a při jejich stisknutí je zobrazeno vyskakovací okno v podobě příspěvku na sociální síti Instagram. Součástí okna je zvětšenina fotografie, profilový obrázek autora, jeho jméno a datum přidání příspěvku. Dále stručný popisek.

Obrázek 27 Modal

4.1.2.3 Patička

Patička je složena ze dvou částí. První část obsahuje seznam odkazů na další stránky webu. Dále textové pole s tlačítkem pro vložení a odeslání emailové adresy k přihlášení odběru novinek. Součástí patičky jsou i odkazy na sociální sítě skrze jejich loga. V druhé části se nachází informace o autorských právech.



Obrázek 28: Patička

Náhled celé stránky je uveden jako příloha 3.

4.1.3 Stránka s detailem produktu

Jedná se o stránku poskytující detailní informace o produktu. Stránka dále obsahuje formulář pro specifikaci objednávaného zboží.

Na stránce se nachází vícero různorodých prvků a celou strukturu lze rozdělit na 3 části, a to hlavičku, hlavní část a patičku, přičemž hlavní část je složena z více menších sekcí. Popis jednotlivých sekcí a prvků je uveden níže.

4.1.3.1 Hlavička a patička

Hlavička a patička je zpracována v identické formě, jak je uvedeno v popisu návrhu úvodní stránky.

4.1.3.2 Hlavní část

Hlavní část je složena z několika stěžejních prvků, které jsou uvedeny níže.

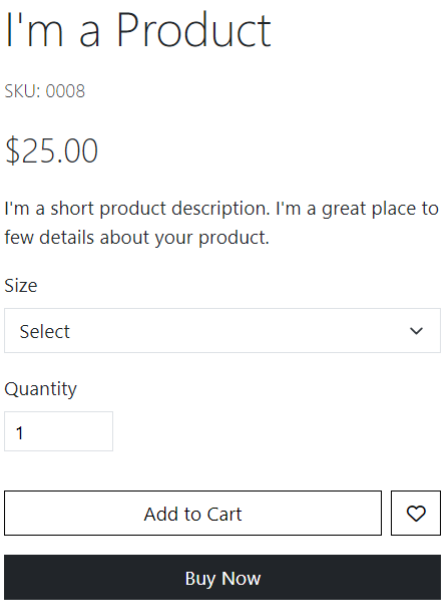
Navigace – Navigace je zde kromě klasické umístěné v hlavičce a patičce stránky realizována taktéž formou drobečkové navigace a tlačítky pro procházení dalších produktů.

Obrázek 29 Navigace u detailu produktu

Produktová fotografie s karuselem – Součástí detailu produktu je pochopitelně i skupina produktových fotografií. Tyto fotografie jsou zobrazeny jednotlivě a změna zobrazené fotografie je prováděna skrze klikání na miniatury dalších fotografií.

Obrázek 30 Produktová fotografie s karuselem

Formulář pro specifikaci objednávky – Kromě názvu produktu, krátkého popisu a ceny se zde nachází výběr velikosti produktu a počet objednávaných kusů, doplněný o 3 CTA tlačítka pro přidání položky do košíku, do oblíbených položek a pro zrychlený nákup.

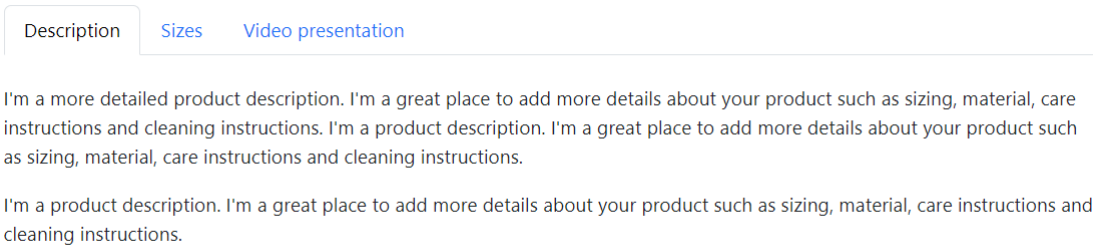


Obrázek 31 Formulář pro specifikaci objednávky

Záložky – Ve spodní části stránky jsou umístěny další, rozsáhlejší informace o produktu. Tyto informace jsou rozděleny do jednotlivých záložek, mezi kterými je možné jednoduše přepínat. Lze tak snadněji dosáhnout na požadovaný obsah bez nutnosti dlouhého rolování po stránce.

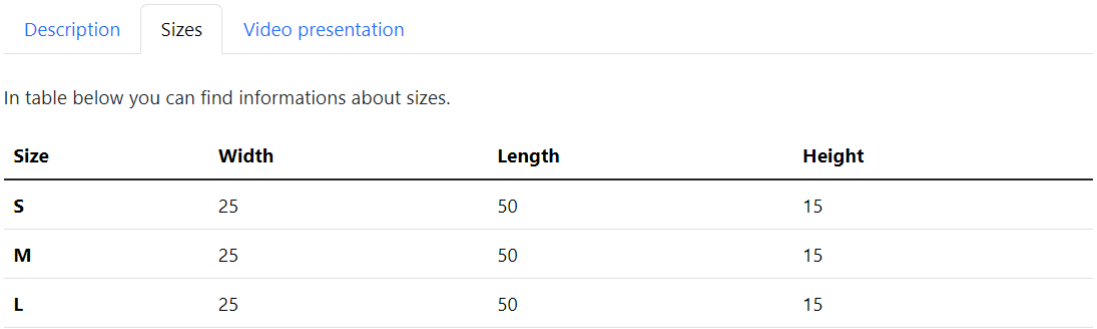
Obrázek 32 Záložky

První záložka – Obsahem první záložky je pouze rozsáhlejší textový popis o produktu.



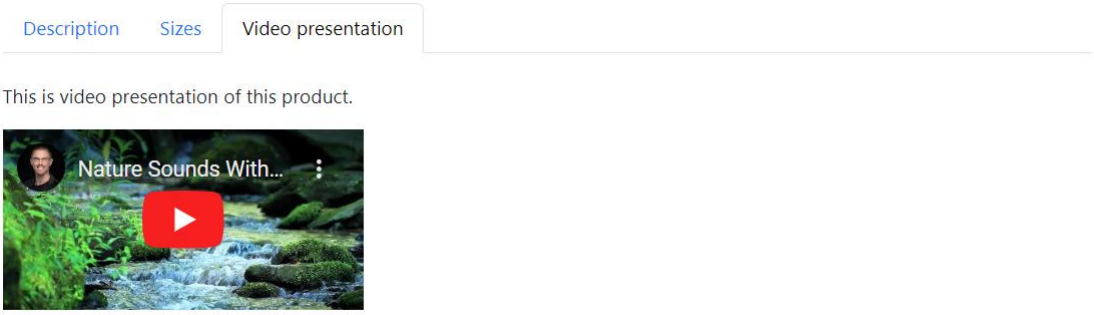
Obrázek 33 První záložka

Druhá záložka – Obsahem druhé záložky je jednoduchá tabulka s údaji o velikostech nabízeného produktu.



Obrázek 34 Druhá záložka

Třetí záložka – Obsahem třetí záložky je produktové video odkazující z platformy YouTube.



Obrázek 35 Třetí záložka

Náhled celé stránky je uveden jako příloha 4.

4.1.4 Stránka s představením společnosti

Jedná se o stránku s jednoduchou prezentací společnosti. Tato stránka má v uživateli vzbudit pocit sympatie a důvěry.

Na stránce se nachází vícero různorodých prvků a celou strukturu lze rozdělit na 3 části, a to hlavičku, hlavní část a patičku, přičemž hlavní část je složena z více menších sekcí. Popis jednotlivých sekcí a prvků je uveden níže.

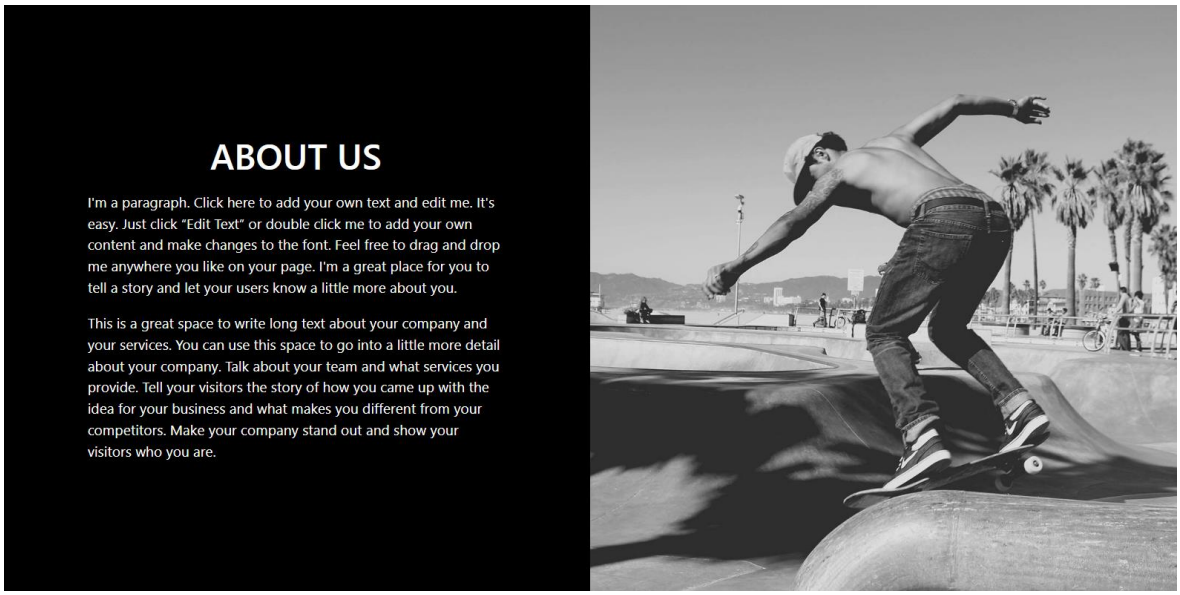
4.1.4.1 Hlavička a patička

Hlavička a patička je zpracována v identické formě, jak je uvedeno v popisu návrhu úvodní stránky.

4.1.4.2 Hlavní část

Hlavní část se skládá ze dvou částí, a to popisem společnosti a uživatelskými recenzemi společnosti.

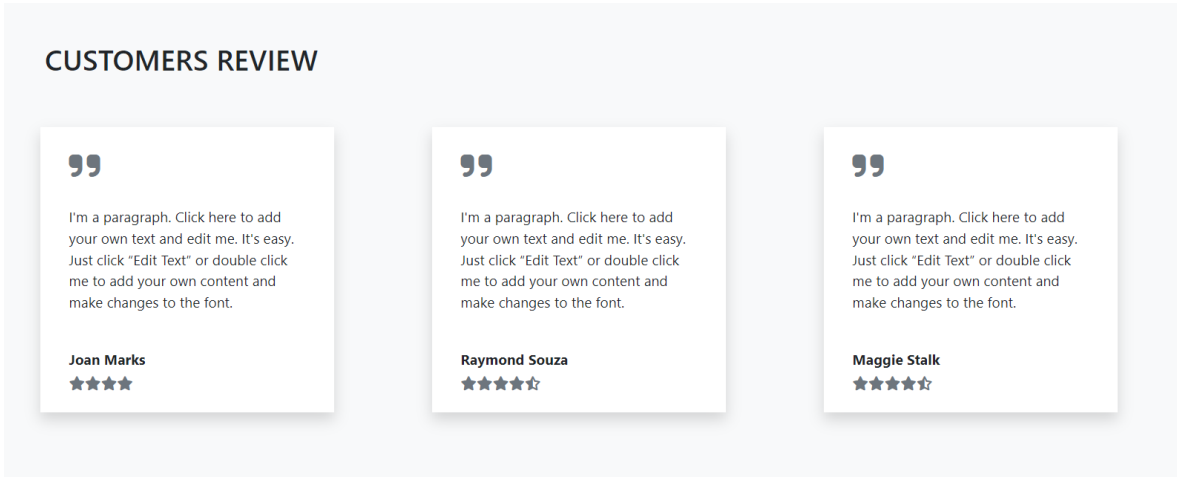
Popis společnosti – Popis společnosti je vyjádřen textovou formou v několika odstavcích. Text je doplněn fotografií pro doplnění a obohacení uživatelského dojmu.



Obrázek 36 Popis společnosti

Recenze – Recenze jsou zpracovány prostřednictvím karet, které díky lehkému stínu vystupují z pozadí, čímž uživatele lépe zaujmou.

Jedná se o úvodní stránku pro blog. Předmětem této stránky je představení tématu blogu a autora uživateli. Jelikož měření probíhá vždy pouze na aktuální stránce, není třeba vytvářet komplexní web.



Obrázek 37 Recenze

Náhled celé stránky je uveden jako příloha 5.

4.2 Výběr CSS frameworků pro testování

Při rozhodování o výběru CSS frameworku je třeba brát do úvahy více faktorů. Primárně by měl být výběr založen na typu projektu, který bude pomocí frameworku

realizován. Dále pak vstupují do rozhodování faktory jako znalost daného frameworku vývojáři realizující projekt, případně křivka učení daného frameworku, možnosti úprav, zajištění přístupnosti, kompatibilita s webovými prohlížeči a kompatibilita s frameworky již používanými v projektu.

Pro výběr CSS frameworku pro tuto práci jsou klíčové následující požadavky:

- Je třeba vytvořit webové stránky z kategorie e-commerce
- Při vývoji by měla být nutnost použití vlastních stylů minimální, v ideálním případě se zcela obejít bez této potřeby
- Všechny vybrané frameworky by měly obsahovat podobné komponenty
- Všechny vybrané frameworky musí disponovat licencí k volnému užití (open-source)
- Všechny vybrané frameworky musí být hojně využívány

Na základě výše uvedených požadavků je upřena pozornost na univerzální frameworky. Díky jejich vlastnostem a komplexitě jsou vhodné k tvorbě různých typů stránek, ze kterých se e-commerce web může skládat, a to bez nutnosti tvorby většího množství vlastních stylů. Dalším důvodem ke zvolení univerzálních frameworků je fakt, že právě tento druh frameworků patří díky svému širokému poli působnosti k nejvíce využívaným.

Výběr frameworků proběhl na základě dat získaných z webového serveru StateOfCSS. Tento web dlouhodobě sleduje popularitu jednotlivých CSS frameworků. Sledované frameworky lze považovat za nejvíce rozšířené. Ze 13 sledovaných frameworků se do kategorie pro obecné použití řadí 5 frameworků. Tyto frameworky byly vybrány pro testování. Jedná se konkrétně o následující CSS frameworky: Bootstrap, Bulma, Fomantic UI, Foundation a Blaze UI.

4.3 Vytvoření experimentálních stránek

Připojení externích CSS a JS souborů samotných frameworků bylo provedeno pomocí CDN odkazu. Tento způsob je mezi vývojáři velmi populární, a tak je použití tohoto způsobu velmi vhodné. V případě přidání vlastních stylů byly vytvořeny a externě připojeny soubory, stejně tak v případě přidání JavaScript kódu. V obou případech byl kód převeden do minifikované podoby, stejně jako je tomu v případě souborů s frameworky.

4.3.1.1 Bootstrap

V případě frameworku Bootstrap nebylo téměř vůbec třeba přidávat vlastní styly ani skripty. Bootstrap je velmi komplexní a poskytuje veškeré prvky pro stavbu běžných stránek. Vlastní úpravy byly třeba pouze v případě tvorby hero sekce na úvodní stránce, kde bylo třeba přidat obrázek jako pozadí. Další úpravy se týkaly specifického vzhledu tlačítka a chování textového pole pro vyhledávání.

4.3.1.2 Bulma

Jelikož Bulma neobsahuje žádný JavaScript kód, bylo třeba potřebné skripty vytvořit a přidat. Potřebný kód se nachází přímo v dokumentaci frameworku, a tak ho bylo možné bez jakýchkoliv úprav převzít, případně byl kód převzat ze stránek W3Schools. Každý skript byl přidán v samostatném souboru, aby bylo možné na stránce načítat pouze skripty, které jsou potřeba.

Úprav z pohledu CSS bylo potřeba více nežli v případě Bootstrapu. Kromě přidání stylů pro vytvoření hero sekce stejně jako u Bootstrapu byly přidány styly pro karusel, úprava tlačítek a odkazů. Jelikož Bulma umožňuje nastavení stylů pro specifické rozměry obrazovky pouze u některých stylů, bylo třeba pro některé případy tuto možnost dodatečně přidat.

4.3.1.3 Fomantic UI

Fomantic UI nabízí množství interaktivních prvků, nicméně využití karuselu nebylo možné a bylo třeba použít kód vlastní. Použitý kód byl stejný jako v případě Bulmy. V případě responzivního menu byla situace obdobná, jelikož framework neumožnil dosáhnout požadovaného chování při přechodu z desktopového zobrazení na zobrazení mobilní, a tak bylo třeba použít vlastní kód, opět stejný jako v případě Bulmy. Pro fungování ostatních skriptů bylo třeba ještě připojit externí knihovnu jQuery.

V případě CSS stylů bylo třeba styly přidat. Kromě přidání vlastní hero sekce bylo třeba přidat styly pro úpravu tlačítek, textu (barvy, velikosti a tučnosti), barvy pozadí, pozicování a navigace. Další nutnou úpravou bylo přidání stylů pro pouze určité velikosti obrazovek.

4.3.1.4 Foundation

Pro správně fungování všech interaktivních prvků bylo třeba připojit externí knihovnu jQuery. Navigace na mobilních telefonech nebyla plně responzivní, a proto byla upravena a opatřena externím JavaScript kódem, který byl použit i u předchozích frameworků. Navigace byla upravena i z hlediska stylů. Ty byly dále rozšířeny o styly pro úpravu tlačítek, textu (barvy, velikosti a tučnosti), barvy pozadí a pozicování. Stejně jako u všech frameworků byla přidána vlastní hero sekce. Dále byly některé styly pro pozicování přidány pro pouze určité velikosti obrazovek.

4.3.1.5 Blaze UI

V případě Blaze UI bylo také třeba použít externí skript pro navigační menu na mobilních zařízeních a pro karusel. Oba skripty se shodují se skripty použitými u dalších frameworků. Úpravy CSS byly třeba pro přidání stylů upravujících tlačítka, text (barvu, velikost, tučnost), barvu pozadí, pozicování a záložek. Některé styly pro pozicování byly navíc upraveny pouze pro určité velikosti obrazovek.

4.4 Výběr metrik

Pro měření výkonu stránek je výběr metrik klíčový. Při špatném výběru metrik nemusí být zjištěny požadované informace a tím pádem nemusí dojít k odhalení slabého místa způsobujícího problém s výkonem stránek. Správným postupem při výběru je tedy především nejprve specifikace cíle měření, tedy jaká informace má být výsledkem měření, a až s touto znalostí je možné provést výběr správných metrik.

Na základě informací z kapitoly 3.7.4 Metriky byly identifikovány jako stěžejní metriky ty v této kapitole obsažené. Tyto metriky pokrývají nejdůležitější oblasti při sledování výkonu webových stránek, a proto je jejich použití klíčové pro splnění cílů práce. Díky zahrnutí většího množství metrik bylo možné důkladněji prověřit zmíněné CSS frameworky a lépe identifikovat, kde se nachází jejich slabiny.

4.5 Zvolení vah pro metriky

Jednotlivé metriky mají pro výsledek výsledného skóre jinou míru významnosti, a proto jsou metriky doplňovány o váhy, které určují, jak velký vliv na celkové skóre mají. Váhy jsou zvoleny tak, aby poskytovaly vyváženou reprezentaci toho, jak uživatel vnímá výkon.

V průběhu času dochází ke změnám vah, aby byl co nejlépe reflektován dopad na výkon vnímaný uživateli. Společnost Google provádí pravidelně průzkum a shromažďuje zpětnou vazbu, a následně upravuje hodnoty vah ve svém nástroji Lighthouse.

Jelikož jsou v této práci sledovány i další metriky, bylo třeba tyto váhy upravit. Změnou jednotlivých vah bylo vytvořeno vícero modelů. V prvním modelu jsou si všechna kritéria rovna. Ve druhém modelu jsou upřednostněny metriky podle doporučení od Google. V tomto případě jsou metriky bez váhy od Google ohodnoceny vahou X. Metriky sledované společností Google jsou ohodnoceny vahou X × váha metriky dle Google. Součet hodnot všech X musí být roven 100. Tato varianta se dále dělí na dvě skupiny. V prvním případě jsou vahou podle Google ohodnoceny pouze metriky měřené v režimu Navigace. Ve druhém případě jsou ohodnoceny metriky měřené v režimu Navigace i v režimu Timespan. Pro obě varianty byly vyhotoveny 3 výstupy, kde byl postupně zvyšován počet X u metrik bez přiřazené váhy společností Google. V prvním výstupu je počet X jedna a v dalších dvou výstupech se zvyšuje na dvě a tři. Poslední model bere všechny předchozí modely a upravuje váhy typům zařízení. Upřednostňováno je mobilní zařízení v poměru 4/3, desktopové zařízení má váhu 2/3.

4.6 Způsob měření

Jak již bylo zmíněno v kapitole 3.7.2.2 Měření metrik, měření lze provádět dvěma způsoby, a to buď v laboratorních podmínkách, nebo sběrem dat od skutečných uživatelů. Ke zvýšení konzistence měření, vyřazením možného zkreslení dat uživateli s výrazně odlišným výkonem jejich zařízení a rychlostí internetového připojení, byla data získávána měřením pouze v laboratorních podmínkách, nikoli ze skutečného provozu stránek.

Měření každé kombinace (konkrétní testovací stránky a konkrétního CSS frameworku) bylo provedeno v sedmi samostatných běžích testu. Tyto testy proběhly

separátně pro mobilní a pro desktopová zařízení. Z naměřených dat byly odebrány případy s nejlepším a nejhorším dosaženým výsledkem.

Jelikož ne veškeré sledované metriky bylo možné měřit při výchozím nastavení programu Lighthouse, kterým je režim Navigace zachycující údaje z prvotního načtení stránky, bylo třeba test každé kombinace rozdělit do dvou částí. Měření druhé části probíhalo v režimu Timespan, který umožňuje uživateli provádět interakce s webovou stránkou, zatímco program Lighthouse sleduje a zaznamenává aktivitu prováděnou na stránce.

4.6.1 Scénáře testování v režimu Timespan

Pro testování v režimu Timespan bylo třeba navrhnout testovací scénář, podle kterého byla řízena aktivita prováděna testovacím uživatelem. Navržený scénář je pro každou stránku, a to jak v desktopové, tak mobilní verzi. Primárním účelem je aktivace interaktivních prvků na stránkách.

4.6.1.1 Úvodní stránka

Desktop – Interakce začíná kliknutím do pole pro vyhledávání umístěného v hlavičce a zadáním libovolného textu. Dalším krokem bylo posunutí na sekci s produkty a kliknutí na tlačítko „Add to Cart“. Poté proběhne opět posun níže na sekci s příspěvky a kliknutí na jeden obrázek pro vyvolání vyskakovacího okna. Okno je dále možno zavřít. Testovací scénář končí posunutím na patičku a vyplněním emailu v poli pro odběr novinek.

Mobil – Scénář pro mobilní zařízení je identický jako pro zařízení desktopová, pouze začíná otevřením navigačního menu pomocí tlačítka a po vyplnění vyhledávacího pole je toto menu opět zavřeno.

4.6.1.2 Detail produktu

Desktop – Interakce začíná stejně jako na úvodní stránce vyplněním vyhledávacího pole. Dalším krokem je změna produktové fotografie pomocí miniatur dalších fotografií. Dále výběr velikosti z rolovacího menu a zvýšení počtu objednávaných produktů. Nakonec kliknutí na tlačítko „Add to Cart“. Dalším krokem je rolování na sekci se záložkami, které je třeba všechny projít. Na závěr třeba opět vyplnit emailovou adresu pro odběr novinek.

Mobil – Scénář pro mobilní zařízení je identický jako pro zařízení desktopová, pouze začíná otevřením navigačního menu pomocí tlačítka a po vyplnění vyhledávacího pole je toto menu opět zavřeno.

4.6.1.3 O společnosti

Na této stránce se nenacházejí žádné další interaktivní prvky, proto jedinou interakcí zůstává vyplnění pole v hlavičce a patičce stránky. V případě mobilního zařízení navíc otevření a zavření menu.

4.6.2 Stolní počítače a mobilní zařízení

Stolní počítače alias „desktopy“ zpravidla disponují vyšším výpočetním výkonem oproti mobilním zařízením. Aby nedocházelo k umělému zlepšování výsledků při měření desktopových zařízení jsou odlišné mezní hodnoty pro rozdělení do intervalů. Desktopová zařízení mají přísnější podmínky pro dosažení dobrého skóre. Pro samotné rozdělení jsou sledovány dva kontrolní body, medián a nejlepších 10 % případů.

Metrika	**Mobilní zařízení**		**Desktop**	
	Medián	P10	Medián	P10
FCP	3000	1800	1600	934
SI	5800	3387	2300	1311
LCP	4000	2500	2400	1200
TBT	600	200	350	150
CLS	0,25	0,1	0,25	0,1
TTI	7300	3785	4500	2468
INP	500	200	500	200
FID	250	130	250	130
TTFB	1800	800	1800	800

Tabulka 2 Hodnoty medián a p10 pro jednotlivé metriky

Z důvodu rozdílného maximálního výkonu desktopových a mobilních zařízení a pravděpodobnosti rychlejšího internetového připojení u desktopových zařízení byly nastaveny odlišné hodnoty takzvaného „throttlingu“. Více o této problematice v kapitole 4.6.3.2 Throttling.

4.6.3 Nástroj Lighthouse

Pro měření hodnot a vyhodnocování výsledků byl zvolen nástroj Lighthouse od společnosti Google. V současné době se jedná o nejrozšířenější nástroj při řešení problematiky měření výkonu webových stránek.

Ačkoli existuje mnoho dalších podobných nástrojů, drtivá většina z nich je odvozena právě z nástroje Lighthouse. Ve většině případů se liší pouze výstupem z auditu (zobrazování dalších ukazatelů) a především se zaměřují na poskytování služeb uživateli. Ať už se jedná o historii auditů, usnadnění pro tvorbu automatizovaných testů nebo dedikovanou podporu.

Lighthouse je komplexní nástroj, který se nestará pouze o průběh testování z aplikačního hlediska, ale také zajišťuje sběr analytických dat, ze kterých jsou odvozovány konstanty pro určité období, které vstupují do průběhu testování a na základě kterých jsou určována výsledná skóre.

Nástroj Lighthouse byl použit ve verzi 11.5.0 v případě spouštění z příkazové řádky. Při použití nástroje přímo v prohlížeči byla použita verze 11.4.0.

4.6.4 Úprava Lighthouse konfigurace

Nástroj Lighthouse nabízí široké možnosti konfigurace, aby se audit testu co nejvíce přizpůsobil požadavkům uživatele. Konfiguraci nástroje Lighthouse lze provádět dvěma způsoby, a to pomocí konfiguračních souborů nebo změnou parametrů. Jelikož bylo cílem získat data ze všech metrik použitých v této práci, nebylo třeba vytvářet vlastní konfigurační soubor. V tomto případě plně postačilo využít výchozí konfigurační soubor programu Lighthouse. Ačkoliv tato konfigurace udává výsledky na základě pouze některých metrik, během testu jsou měřeny všechny a potřebná data lze nalézt v konečném výpisu a provést s nimi vlastní výpočet vyhodnocení. Změny parametrů testu pomocí klíčových atributů bylo naopak využito hojně.

4.6.4.1 Mobilní a desktopová zařízení

Jak již bylo řečeno, testy byly prováděny na dvou typech zařízení (mobilní a desktopové). Pro specifikaci, o jaký druh se jedná, je třeba příkaz v příkazové řádce rozšířit o atributy nesoucí tuto informaci. Dále je třeba specifikovat rozlišení zařízení.

Podle serveru Altamira.ai je nejvíce rozšířeným rozlišením obrazovky u desktopových zařízení 1366×768 a u mobilních zařízení 360×640.

Parametry specifikaci zařízení a rozlišení pro mobilní zařízení:

--form-factor=mobile

--screenEmulation.mobile

--screenEmulation.width=360

--screenEmulation.height=640

Parametry specifikaci zařízení a rozlišení pro desktopová zařízení:

--preset=desktop

--form-factor=desktop

--screenEmulation.mobile=false

--screenEmulation.width=1366

--screenEmulation.height=768

4.6.4.2 Throttling

Využití throttlingu je důležitou součástí testování. Throttling znamená proces cíleného omezování zdrojů. Nastavením hladiny maximálního využití zdrojů tak, aby jich bylo vždy dosaženo, je zapříčiněn konzistentnější a lépe kontrolovaný průběh načítání stránky. Nastavením throttlingu na hodnoty odpovídající průměrným, či až podprůměrným zdrojům je dosaženo lepší vypovídající hodnoty dat. Právě v případech, kdy je k dispozici nedostatek zdrojů se nejlépe odráží problémy s rychlostí načítání stránek. Throttling je prováděn zpravidla pro omezování rychlosti internetového připojení a omezování výkonu procesoru zařízení.

Způsob provedení throttlingu rychlosti internetového připojení bylo vybráno simulované omezení. Jedná se o výchozí omezení v nástroji Lighthouse a oproti způsobu omezení na úrovni požadavku (též označováno jako omezení DevTools) dosahuje obecně přesnějších výsledků. U throttlingu síťového připojení lze kromě způsobu provedení nastavit 4 další parametry, a to RTT (odezva), síťovou propustnost stahování, síťovou propustnost nahrávání a ztrátu paketů. Čím je rychlost a kvalita připojení nižší, tím lépe odráží kvalitu výkonnosti stránek. Pro mobilní zařízení jsou výchozí hodnoty nastaveny následovně: odezva – 150ms, propustnost stahování – 1,6Mbps, propustnost nahrávání – 750Kbps, ztráta paketů – žádná. Tyto hodnoty mají dobrou vypovídající hodnotu pro účely

této práce, pouze rychlost stahování byla snížena na 1,5Mbps. U desktopových zařízení jsou výchozí hodnoty následující: odezva – 40ms, propustnost stahování 10,24Mbps, propustnost nahrávání – 1,024Mbps a ztráta paketů – žádná. Tyto hodnoty jsou ovšem příliš dobré, a tak by mohlo dojít ke stavu, kdy by byly všechny naměřené hodnoty hodnoceny plným počtem bodů. Z tohoto důvodu byly sníženy na následující hodnoty: odezva – 100ms, propustnost stahování – 5Mbps a propustnost nahrávání – 1,5Mbps.

Parametry pro změnu throttlingu síťového připojení mobilních zařízení:

--throttling-method=simulate

--throttling.rttMs=150

--throttling.requestLatencyMs=150

--throttling.throughputKbps=1500

--throttling.downloadThroughputKbps=1500

--throttling.uploadThroughputKbps=750

Parametry pro změnu throttlingu síťového připojení desktopových zařízení:

--throttling-method=simulate

--throttling.rttMs=100

--throttling.requestLatencyMs=100

--throttling.throughputKbps=5000

--throttling.downloadThroughputKbps=5000

--throttling.uploadThroughputKbps=1500

Nastavení throttlingu procesoru funguje na mírně odlišném principu. Zde není možné nastavit hodnotu udávající množství zdrojů explicitně. Pro omezení výkonu procesoru je užíváno omezení násobiče procesoru. V závislosti na maximálním výkonu procesoru a na požadavku pro omezení simulující výkon procesoru z nižší výkonnostní kategorie je udáván koeficient. Čím je vyšší číslo koeficientu, tím vyšší je míra throttlingu. U každého reportu z auditu je uvedeno výkonnostní skóre zařízení, na kterém byl test prováděn. U zařízení používaného v této práci se hodnota pohybuje v rozmezí 1900-2000. Z tabulky níže je patrné, že se toto zařízení řadí do kategorie „high-end desktop“. V této tabulce je také uveden přibližný koeficient rozdílu mezi high-end desktop zařízením a méně výkonnými zařízeními.

	High-End Desktop	Low-End Desktop	High-End Mobile	Mid-Tier Mobile	Low-End Mobile
Lighthouse BenchmarkIndex	1500-2000	1000-1500	800-1200	125-800	<125
Násobitel	1x	2x (1-4)	2x (1-4)	4x (2-10)	10x (5-20)

Tabulka 3 Lighthouse BenchmarkIndex a násobitel

Hodnota před závorkou označuje navrhovaný násobitel. Pokud se skóre zařízení blíží více jednomu či druhému konci rozsahu, je vhodné násobitel upravit podle rozsahu uvedeném v závorce. Jelikož se výkon testovacího zařízení pohybuje na horní hranici high-end desktop zařízení, použité koeficienty byly zvoleny z horní hranice. V případě mobilního i desktop zařízení je cílem simulovat low-end zařízení. V případě mobilního zařízení tak byl použit koeficient 15 a v případě desktop zařízení koeficient 4.

Parametry pro změnu throttlingu procesoru pro simulaci Low-End mobilních zařízení:

--throttling.cpuSlowdownMultiplier=15

Parametry pro změnu throttlingu procesoru pro simulaci Low-End desktopových zařízení:

--throttling.cpuSlowdownMultiplier=4

	Mobilní zařízení	Desktop
CPU	15	4
Odezva	150	100
Propustnost stahování	1500	5000
Propusnost nahrávání	750	1500

Tabulka 4 Hodnoty pro throttling

4.6.5 Výběr a konfigurace prohlížeče

Při měření prostřednictvím nástroje Lighthouse spouštěného z příkazové řádky je třeba testovanou stránku otevřít ve webovém prohlížeči, ve kterém následně proběhne test. Pro měření byl vybrán webový prohlížeč Google Chrome, jelikož je v případě webového prohlížeče i nástroje Lighthouse stejný vývojář, a to společnost Google. Díky tomu byla zajištěna maximální kompatibilita.

Webový prohlížeč Google Chrome byl aktualizován na aktuálně nejnovější verzi 122.0.6261.58 v 64-bit verzi. Aby bylo zamezeno zkreslení měření, ať už kvůli ukládání dat z testovacích stránek či kvůli uživatelskému nastavení byla v případě spouštění testu z příkazové řádky spouštěna instance prohlížeče s novým uživatelským profilem a při testování v Timespan módů nová instance anonymního režimu. Obě tyto metody dosahují stejných výsledků. Takto nastavený prohlížeč taktéž automaticky deaktivuje veškerá rozšíření prohlížeče.

4.6.6 Hosting experimentálních stránek

Aby mohlo být prováděno měření prostřednictvím příkazové řádky, bylo třeba umístit experimentální webové stránky na webový server. Dále bylo třeba stránky uveřejnit pod skutečnou doménou. Hosting byl zajištěn poskytovatelem webzdarma.cz s balíčkem základ. Tento balíček obsahuje SSL/HTTPS šifrování a rychlé SSD disky. Tímto je simulováno standardní chování webového serveru.

4.7 Výpočet výsledného skóre

Výsledkem měření je vypočítané skóre pro každou metriku. Toto skóre je automaticky vypočítáváno nástrojem Lighthouse. Z nasbíraných dat z vícero opakování měření je vypočtena průměrná hodnota, která je následně násobena váhou pro konkrétní metriku, označována jako index. Tyto hodnoty jsou zprůměrovány a ukazují skóre konkrétní stránky na konkrétním zařízení.

Pro výpočet celkového skóre jedné stránky ještě dochází k vytvoření průměru z vypočtených indexů pro mobilní a desktopovou variantu. V případě prioritizace mobilního zařízení jsou tyto indexy násobeny vahami.

Celkové konečné skóre pro daný váhový model je pak vypočítáno jako průměr celkových skóre ze všech experimentálních stránek.

5 Výsledky a diskuse

Tato kapitola se věnuje interpretaci dat a jejich vyhodnocením. Výsledné skóre nabývá hodnot z intervalu <0;1>, kdy platí, čím vyšší hodnota, tím lepší výsledek.

5.1 Analýza dat

V naměřených datech se nachází určité vzory. Prvním vzorem je, že průměrný výsledek ze všech metrik dosahuje lepších hodnot u desktopových zařízení. Další vzor lze nalézt při pohledu na průměrné výsledky z jednotlivých experimentálních stránek. Zde je dosahováno konstantně nejlepších výsledků na stránce s popisem společnosti, na druhém místě se nachází úvodní stránka a nejhorších výsledků je dosahováno na stránce s detailem produktu. Výjimkou je pouze framework Fomantic UI, u kterého je nejhorších výsledků dosaženo na úvodní stránce.

Obecně nejhorších výsledků napříč všemi frameworky je dosahováno u metrik First Contentful Paint a Largest Contentful Paint. Naopak nejlepších výsledků je dosahováno u metriky Time To First Byte, kde výsledky přesahují hranici indexu 0,9.

5.2 Stejné váhy

V případě, že mají všechny sledované metriky stejné váhy je z výsledů patrné, že rozdíly mezi prvními čtyřmi nejlépe hodnocenými frameworky nejsou zásadní. Nejhůře dopadl framework Fomantic UI, u něhož je zhoršení skóre oproti ostatním značné.

	Skóre	*Pořadí*
Bootstrap	0,89	1
Bulma	0,88	2
Fomantic UI	0,69	5
Foundation	0,84	4
Blaze UI	0,86	3

Tabulka 5 Výsledky měření se stejnými vahami – vyšší hodnota skóre = lepší

Pokud je kladena vyšší váha na výsledky mobilních zařízení, na výsledcích se to nijak neprojeví. K žádné dramatické změně hodnot nedošlo, pouze k drobnému zhoršení v řádu jednotek setin. Ke zhoršení výsledků došlo rovnoměrně napříč všemi frameworky, a tak se pořadí oproti rovnosti vah mezi desktopovými a mobilními zařízeními nezměnilo.

	Skóre	*Pořadí*
Bootstrap	0,89	1
Bulma	0,87	2
Fomantic UI	0,67	5
Foundation	0,83	4
Blaze UI	0,85	3

Tabulka 6 Výsledky měření se stejnými vahami s preferencí mobilního zařízení – vyšší hodnota skóre = lepší

5.3 Váhy podle Google

Výsledky při použití vah inspirovanými společností Google neobsahují velké rozdíly, avšak v některých případech ke změnám v pořadí došlo. Podrobněji rozepsané výsledky jsou uvedeny níže.

5.3.1 Váhy podle Google u metrik měřených výchozím způsobem

Výsledné hodnoty jsou si velmi podobné napříč všemi třemi ohodnoceními metrik. Lze zde nalézt trend zlepšování výsledného skóre, avšak rozdíly mezi jednotlivými frameworky zůstávají v podstatě identické.

Oproti modelu, kdy byly všechny metriky ohodnoceny stejnými vahami zde došlo ke změně mezi prvními dvěma frameworky. Nejlepší skóre tak dosahuje Bulma a na druhém místě Bootstrap.

Rozdíly jsou zde taktéž větší. Ačkoli rozdíl mezi nejlepším a nejhorším skóre zůstal zachován, frameworky se rozdělily do pomyslných 3 skupin. Rozdíl mezi těmito skupinami je znatelný, ale rozdíl uvnitř těchto skupin je zanedbatelný. První skupina je tvořena frameworky Bulma a Bootstrap, druhá skupina frameworky Blaze UI a Foundation a do poslední skupiny spadá framework na posledním pátém místě Fomantic UI.

	1X		*2X*		*3X*	
	Skóre	Pořadí	Skóre	Pořadí	Skóre	Pořadí
Bootstrap	0,85	2	0,85	2	0,86	2
Bulma	0,87	1	0,87	1	0,88	1
Fomantic UI	0,68	5	0,69	5	0,69	5
Foundation	0,78	4	0,78	4	0,79	4
Blaze UI	0,80	3	0,81	3	0,82	3

Tabulka 7 Výsledky měření s vahami podle Google v režimu Navigace – vyšší hodnota skóre = lepší

V případě upřednostnění výsledků z mobilních zařízení se stejně jako v případě stejných vah pro všechny metriky nijak nemění oproti výsledkům s rovnoměrnými vahami mezi desktopovými a mobilními zařízeními. Ani zde se výsledky téměř neliší a pořadí

zůstává stále stejné, tedy na prvním místě framework Bulma, následuje Bootstrap, Blaze UI, Foundation a na posledním místě Fomantic UI.

	1X		*2X*		*3X*	
	Skóre	Pořadí	Skóre	Pořadí	Skóre	Pořadí
Bootstrap	0,85	2	0,86	2	0,86	2
Bulma	0,87	1	0,87	1	0,87	1
Fomantic UI	0,66	5	0,67	5	0,67	5
Foundation	0,76	4	0,77	4	0,78	4
Blaze UI	0,80	3	0,81	3	0,81	3

Tabulka 8 Výsledky měření s vahami podle Google v režimu Navigace s preferencí mobilního zařízení – vyšší hodnota skóre = lepší

5.3.2 Váhy podle Google u metrik měřených způsoby Navigace i Timespan

V případě přiřazení větší váhy i metrikám v režimu Timespan lze pozorovat opět větší vyrovnanost mezi nejlepšími čtyřmi frameworky. Blaze UI se zde výrazněji posunul a dosáhl na stejný výsledek jako framework Bulma a tím se dostal na dělenou druhou pozici. Výsledky se více podobají modelu se stejnými vahami pro všechny sledované metriky. Nejlepšího hodnocení zde dosáhl framework Bootstrap následovaný frameworky Bulma a Blaze UI. S těsným rozdílem následuje Foundation a na posledním místě se s markantnějším rozdílem opět umístil Fomantic UI.

	1X		*2X*		*3X*	
	Skóre	Pořadí	Skóre	Pořadí	Skóre	Pořadí
Bootstrap	0,89	1	0,89	1	0,89	1
Bulma	0,86	2	0,87	2	0,87	2
Fomantic UI	0,71	5	0,71	5	0,71	5
Foundation	0,84	4	0,84	4	0,85	4
Blaze UI	0,86	2	0,87	2	0,87	2

Tabulka 9 Výsledky měření s vahami podle Google v režimech Navigace i Timespan – vyšší hodnota skóre = lepší

S preferencí mobilních zařízení došlo k drobné změně, kdy přestaly být frameworky Bulma a Blaze UI na stejné úrovni, ale Blaze UI dosáhl nepatrně lepšího výsledku a odsunul framework Bulma na třetí příčku. S nejlepším skóre se první příčky opět ujal Bootstrap, čtvrté místo obsadil Foundation a na posledním místě zůstal Fomantic UI.

	1X		2X		3X	
	Skóre	Pořadí	Skóre	Pořadí	Skóre	Pořadí
Bootstrap	0,89	1	0,89	1	0,89	1
Bulma	0,85	3	0,85	3	0,85	3
Fomantic UI	0,69	5	0,69	5	0,69	5
Foundation	0,83	4	0,83	4	0,83	4
Blaze UI	0,86	2	0,86	2	0,86	2

Tabulka 10 Výsledky měření s vahami podle Google v režimu Navigace i Timespan s preferencí mobilního zařízení – vyšší hodnota skóre = lepší

5.4 Souhrnné zhodnocení

Použitím více váhových modelů bylo možné naměřená data komplexněji vyhodnotit. Po prozkoumání výsledků ze všech váhových modelů lze stanovit následující tvrzení. Ačkoliv dosahují výsledky z mobilních zařízení horších výsledků nežli ze zařízení desktopových, zvýšením váhy pro mobilní zařízení nedošlo k citelným změnám ve výsledcích ani v pořadí frameworků. Pořadí frameworků se napříč všemi použitými modely v některých případech liší, avšak lze sledovat, že některé frameworky dosahují dobrých výsledků nehledě na použitý váhový model, a naopak některé frameworky dosahují výsledků špatných.

Frameworky Bootstrap a Bulma se pravidelně umísťovaly s nejlepším hodnocením na předních příčkách. Díky dobrým výsledkům napříč všemi sledovanými metrikami lze tvrdit, že ani další změna vah neovlivní kvalitu výsledků. Tyto frameworky jsou doporučeny k použití v případě zaměření na rychlost načítání webového obsahu.

Frameworky Blaze UI a Foundation byly méně konzistentní při změnách vah. V případech stejných vah pro všechny metriky a vah podle Google měřených způsoby Navigace i Timespan bylo dosaženo lepších výsledků, které se více přibližovaly frameworkům Bootstrap a Bulma. V případě vah dle Google dosáhl framework Blaze UI dokonce lepších výsledků nežli Bulma. Tyto frameworky lze při výběru frameworku stále označit za možnou volbu.

S nejhoršími výsledky skončil framework Foundation UI. Zde bylo dosahováno konstantně nejhorších výsledků se značným rozdílem. Ačkoliv tyto hodnoty, stejně jako u zbylých frameworků, stále spadají do kategorie „Potřebuje zlepšení“ a ne do kategorie „Špatné“, použití tohoto frameworku není doporučeno. Rychlost načítání webového obsahu zde může výrazně ovlivnit uživatelskou zkušenost s používáním webových stránek a řadit stránky ve vyhledávači na nižší pozice.

Framework	Doporučení
Bootstrap	Doporučeno
Bulma	Doporučeno
Blaze UI	Použitelné
Foundation	Použitelné
Fomantic UI	Nedoporučeno

Tabulka 11 Souhrnné výsledky

Zdrojová data jsou uvedena v příloze 1.

6 Závěr

Teoretická část této práce byla zpracována na základě analýzy a následné syntézy vědeckých a odborných informačních zdrojů zabývajících se primárně tématem vývoje webových stránek, CSS frameworky, metrikami pro měření rychlosti načítání webového obsahu a přehledem obdobných řešení.

Pro řešení dílčích cílů práce bylo klíčové studium informačních zdrojů. Především v případě charakterizace využití CSS frameworků a technologií pro jejich vývoj a při vytváření přehledu obdobných řešení a doporučení. V případě tvorby experimentálních stránek bylo třeba nejprve vytvořit návrh, který bude obsahovat širokou paletu prvků a bude tak pokrývat většinu prvků, se kterými se lze na webových stránkách setkat. Současně byl kladen důraz na použití takových prvků, které jsou uvedeny v dokumentaci pro všechny frameworky, tudíž nedojde ke snížení objektivnosti měření. Samotná realizace webových stránek byla založena na znalosti HTML a studiem dokumentace jednotlivých CSS frameworků pro správné použití stylů. Volba kritérií reflektuje nejvíce používané metriky sledující rychlost načítání webového obsahu. Některé metriky mohou být považovány v dnešní době za již méně významné, tomu jsou přizpůsobeny jejich váhy.

Hlavním cílem práce bylo srovnání vybraných CSS frameworků s obdobnou charakteristikou použití. Srovnání bylo zaměřeno na rychlost načítání webového obsahu. Navržením experimentálních stránek a jejich realizací pomocí vybraných frameworků bylo možné frameworky otestovat podle zvolených kritérií, které byly doplněny o váhy v několika váhových modelech. Pomocí nástroje Lighthouse od společnosti Google byla nasbírána potřebná data. Na základě těchto dat bylo možné stanovit závěry.

Testování bylo podrobeno celkem 5 CSS frameworků s obecným charakterem využití, kterými byly vytvořeny 3 experimentální webové stránky z oblasti e-commerce. Měřením a následným průzkumem a vyhodnocením dat bylo dosaženo zjištění, které CSS frameworky z testování v této práci jsou vhodné pro použití s ohledem na rychlost načítání webového obsahu. Doporučit lze frameworky Bootstrap a Bulma, které se stabilně pohybovaly na předních příčkách. I v rámci výsledků jednotlivých metrik bylo dosahováno ve všech případech velmi vysokého hodnocení. Opačným případem je framework Fomantic UI, který v hodnocení výrazně ztrácel na další testované frameworky. Výsledky zde byly spíše průměrné a v případě některých metrik vyloženě špatné. Z tohoto důvodu

není doporučeno užití frameworku Fomantic UI v případě, kdy je kladen důraz na rychlost načítání webového obsahu.

Při průzkumu dat bylo dále zjištěno, že i přes použití výrazného throttlingu byla naměřená data z mobilního zařízení obecně horší nežli ze zařízení desktopového. Tento fakt podporuje užití váhového modelu ve prospěch právě mobilních zařízení. Ačkoliv mobilní zařízení v tomto případě disponuje vyšší váhou, na výsledném pořadí frameworků se neprojevila žádná změna.

Tato práce přináší přehledné srovnání CSS frameworků se zaměřením na rychlost načítání webového obsahu a poskytuje cenné informace pro podporu rozhodování při výběru frameworku.

Možnost rozšíření práce představuje především zahrnutí do srovnání více frameworků a také frameworků z dalších skupin. V takovém případě by mohlo dojít k zajímavému zjištění, jaký je rozdíl v rychlosti načítání webového obsahu s použitím různých druhů frameworků, speciálně v porovnání s frameworky zaměřenými právě na tuto problematiku.

7 Seznam použitých zdrojů

1. *Fast load times* [online], c2023. [cit. 2023-12-01]. Dostupné z: https://web.dev/explore/fast
2. *Less* [online], c2023. [cit. 2023-12-01]. Dostupné z: https://lesscss.org/#
3. *Sass* [online], c2006-2023. [cit. 2023-12-01]. Dostupné z: Sass Basics
4. ABDAAL, Taimur, 2019. Typography for Developers. In: *CSS-TRICKS* [online]. [cit. 2023-12-01]. Dostupné z: https://css-tricks.com/typography-for-developers/
5. ADARSH, Gupta, 2023. 24 Best CSS Frameworks To Look Forward In 2023. In: *Lambdatest* [online]. [cit. 2023-12-01]. Dostupné z: https://www.lambdatest.com/blog/best-css-frameworks/
6. AJMI, Ahmad, 2014. Using Helper Classes to DRY and Scale CSS. In: *Sitepoint* [online]. [cit. 2023-12-01]. Dostupné z: https://www.sitepoint.com/using-helper-classes-dry-scale-css/
7. AMOLLO, Benson, 2023. How do you customize and extend a CSS framework without breaking its functionality? In: *Linkedin* [online]. [cit. 2023-11-08]. Dostupné z: https://www.linkedin.com/advice/0/how-do-you-customize-extend-css-framework
8. ANSTEY, Chris a Bojan Pavic, 2019. What is speed? In: *Web.dev* [online]. [cit. 2023-12-01]. Dostupné z: https://web.dev/articles/what-is-speed
9. BATOOL, Farah, 2023. JavaScript: Client-side vs Server-side. In: *Linuxhint* [online]. [cit. 2023-12-01]. Dostupné z: https://linuxhint.com/javascript-client-side-vs-server-side/
10. BIN UZAYR, Sufyan, 2022. *Web Performance Optimization* [online]. Taylor & Francis Group [cit. 2023-11-13]. ISBN 9781000541342. Dostupné z: https://ebookcentral.proquest.com/lib/czup/detail.action?docID=6868919
11. BISHOP, Russell, 2019. CSS utility classes: Your library of extendable styles. In: *LogRocket* [online]. [cit. 2023-12-01]. Dostupné z: https://blog.logrocket.com/css-utility-classes-library-extendable-styles/
12. Core Web Vitals — A Google Initiative to measure performance, 2024. In: *Medium* [online]. [cit. 2024-03-16]. Dostupné z: https://yuvrajpy.medium.com/core-web-vitals-%EF%B8%8Fa-google-initiative-to-measure-performance-130a189cecfd

13. CSS flexible box layout, 2023. In: *Mdn web docs* [online]. [cit. 2023-12-01]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_flexible_box_layout
14. CSS grid layout, 2023. In: *Mdn web docs* [online]. [cit. 2023-12-01]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_grid_layout
15. DI NOIA, Tommaso, In-Young KO, Markus SCHEDL a Carmelo ARDITO, 2022. *Web Engineering*. Berlin: Springer. ISBN 978-3-031-09916-8.
16. Document and website structure, 2023. In: *Mdn web docs* [online]. [cit. 2023-11-17]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Document_and_website_structure
17. ENE ANYEBE, Blessing, 2023. A Guide to Optimizing JavaScript Files. In: *Sitepoint* [online]. [cit. 2023-11-16]. Dostupné z: https://www.sitepoint.com/optimizing-javascript-files/
18. FRIEDMAN, Vitaly, 2018. *Responsive Web Design: What It Is And How To Use It* [online]. [cit. 2023-11-07]. Dostupné z: https://www.smashingmagazine.com/2011/01/guidelines-for-responsive-web-design/
19. GADHIYA, Mehul, 2023. Step-by-Step Tutorial to Cross Browser Compatibility. In: *LAMBDATEST* [online]. [cit. 2023-12-01]. Dostupné z: https://www.lambdatest.com/learning-hub/cross-browser-compatibility
20. Harnessing The Power Of Top 15 CSS Frameworks: Responsive Web Design Unleashed, 2023. In: *Nestify* [online]. [cit. 2023-11-18]. Dostupné z: https://nestify.io/blog/top-css-frameworks/
21. HARRIS, Andy, 2014. *HTML5 and CSS3 All-in-One For Dummies*. 3rd Edition. New Jersey: John Wiley. ISBN 978-1-118-41983-0.
22. HAZEEZ, Habdul, 2022. Best practices for improving CSS performance. In: *LogRocket* [online]. [cit. 2023-11-16]. Dostupné z: https://blog.logrocket.com/best-practices-improving-css-performance/
23. HERNANDEZ, Ian, 2023. 16 Popular CSS Frameworks That Will Help You Save Time (With Style). In: *DreamHost* [online]. [cit. 2023-12-01]. Dostupné z: https://www.dreamhost.com/blog/css-frameworks/
24. HILL, Tim, 2023. 20+ Essential Parts of a Website [Updated]. In: *SEOptimer* [online]. [cit. 2023-11-17]. Dostupné z: https://www.seoptimer.com/blog/parts-of-a-website/

25. HOWE, Shay, 2014. *LEARN TO CODE HTML & CSS: DEVELOP & STYLE WEBSITES*. Indianapolis: New Riders. ISBN 978-0-321-94052-0.
26. KARLSSON, Jill, 2014. *Responsive web design with CSS frameworks* [online]. [cit. 2023-03-28]. ISSN: 1401-5749.
27. MIHAJLIJA, Milica a Philip WALTON, 2023. Cumulative Layout Shift (CLS). In: *Web.dev* [online]. [cit. 2023-12-02]. Dostupné z: https://web.dev/articles/cls
28. MONUS, Anna, 2023. Popular CSS preprocessors with examples: Sass, Less, Stylus and more. In: *RAYGUN* [online]. [cit. 2023-12-01]. Dostupné z: https://raygun.com/blog/css-preprocessors-examples/
29. POLLARD, Barry a Philip WALTON, 2023. Largest Contentful Paint (LCP). In: *Web.dev* [online]. [cit. 2023-12-01]. Dostupné z: https://web.dev/articles/lcp
30. POWELL, Thomas A., 2010. *HTML & CSS: The Complete Reference*. Fifth Edition. New York: McGraw-Hill. ISBN 978-0-07-174170-5.
31. PRAJAPATI, Lokesh, 2023. The JavaScript Optimization Checklist: 7 Things You Need to Do to Speed Up Your Website. In: *Medium* [online]. [cit. 2023-11-16]. Dostupné z: https://lokesh-prajapati.medium.com/the-javascript-optimization-checklist-7-things-you-need-to-do-to-speed-up-your-website-be24873ef2da
32. SASS Vs SCSS: What's The Difference?, 2023. In: *InterviewBit* [online]. [cit. 2023-12-01]. Dostupné z: https://www.interviewbit.com/blog/sass-vs-scss/
33. SEMAH, Benjamin, 2023. Top 20 Best CSS Frameworks for Front-End Developers in 2023. In: *Hackr.io* [online]. [cit. 2023-12-01]. Dostupné z: https://hackr.io/blog/best-css-frameworks
34. SEWELL, Steve, 2022. The Complete Guide to Optimizing CSS for Fast Page Loads. In: *Builder.io* [online]. [cit. 2023-11-16]. Dostupné z: https://www.builder.io/blog/the-complete-guide-to-optimizing-css-for-fast-page-loads
35. SHENOY, Aravind a Anirudh PRABHU, 2018. *CSS Framework Alternatives: Explore Five Lightweight Alternatives to Bootstrap and Foundation with Project Examples*. Mumbai: Springer. ISBN 978-1-4842-3398-6.
36. SHIVAKUMAR, Shailesh Kumar, 2020. *Modern Web Performance Optimization: Methods, Tools, and Patterns to Speed Up Digital Platforms*. Bengaluru: Apress. ISBN 978-1-4842-6528-4.

37. Speed Index, 2024. In: *Mdn web docs* [online]. [cit. 2024-03-16]. Dostupné z: https://developer.mozilla.org/en-US/docs/Glossary/Speed_index
38. UNADKAT, Jash, 2023. Mobile First Design: What it is and How to Implement it. In: *BrowserStack* [online]. [cit. 2023-11-07]. Dostupné z: https://www.browserstack.com/guide/how-to-implement-mobile-first-design
39. WAGNER, Jeremy a Barry POLLARD, 2021. Time to First Byte (TTFB). In: *Web.dev* [online]. [cit. 2023-12-02]. Dostupné z: https://web.dev/articles/ttfb
40. WAGNER, Jeremy, 2023. Interaction to Next Paint (INP). In: *Web.dev* [online]. [cit. 2023-12-02]. Dostupné z: https://web.dev/articles/inp
41. WALTON, Philip, 2019. First Contentful Paint (FCP). In: *Web.dev* [online]. [cit. 2023-12-01]. Dostupné z: https://web.dev/articles/fcp
42. WALTON, Philip, 2019. User-centric performance metrics. In: *Web.dev* [online]. [cit. 2023-12-01]. Dostupné z: https://web.dev/articles/user-centric-performance-metrics
43. WALTON, Philip, 2023. First Input Delay (FID). In: *Web.dev* [online]. [cit. 2023-12-02]. Dostupné z: https://web.dev/articles/fid
44. WALTON, Philip, 2023. Time to Interactive (TTI). In: *Web.dev* [online]. [cit. 2023-12-02]. Dostupné z: https://web.dev/articles/tti
45. WALTON, Philip, 2023. Total Blocking Time (TBT). In: *Web.dev* [online]. [cit. 2023-12-02]. Dostupné z: https://web.dev/articles/tbt
46. What is JavaScript?, 2023. In: *Mdn web docs* [online]. [cit. 2023-12-01]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript
47. WOOD, Adam, c2015-2023. HTML5 Basics For Everyone Tired Of Reading About Deprecated Code. In: *Html.com* [online]. [cit. 2023-11-30]. Dostupné z: https://html.com/html5/
48. YALANSKA, Marina, 2023. The Anatomy of a Web Page: 14 Basic Elements. In: *Tubikblog* [online]. [cit. 2023-11-17]. Dostupné z: https://blog.tubikstudio.com/anatomy-of-web-page/

8 Seznam obrázků, tabulek, grafů a zkratek

8.1 Seznam obrázků

8.2 Seznam tabulek

8.3 Seznam použitých zkratek

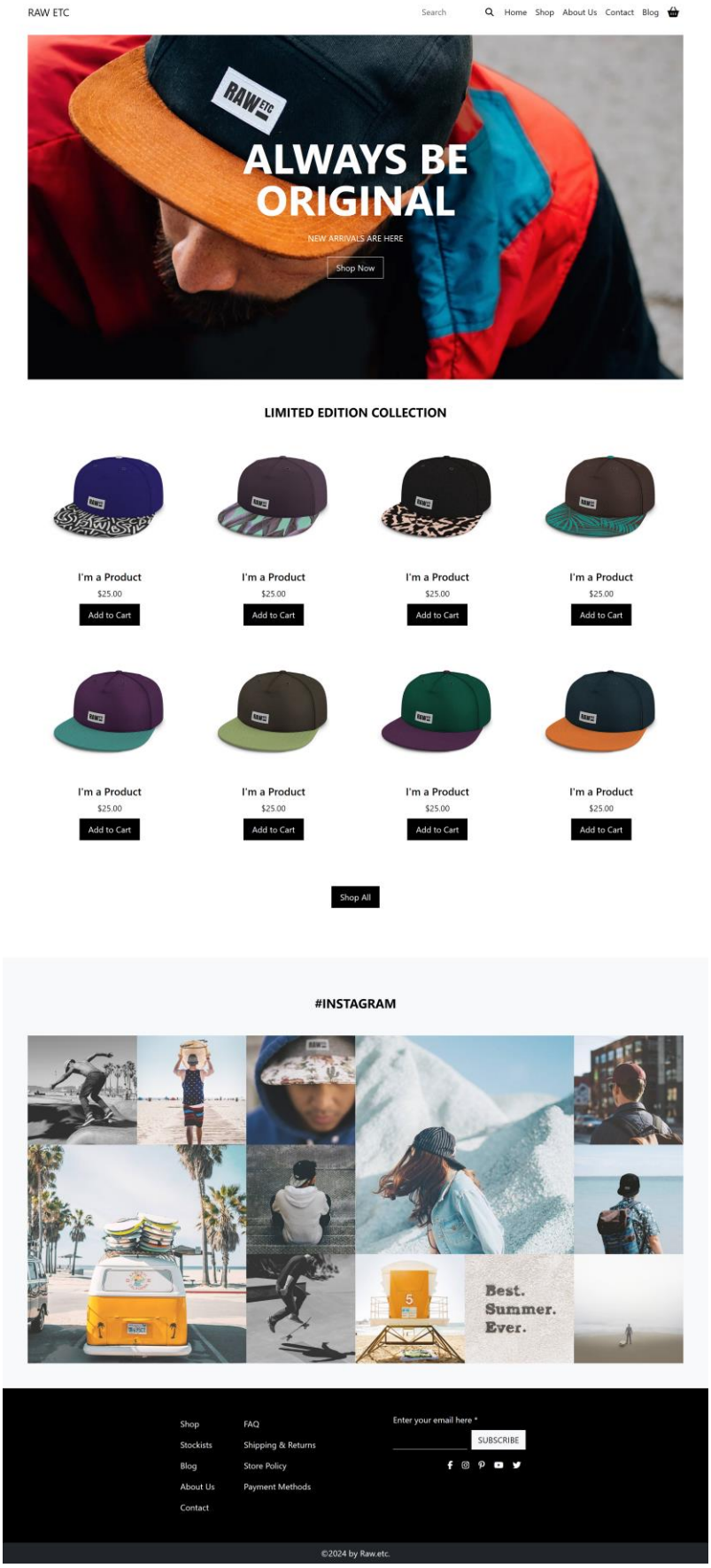
HTML – Hypertext Markup Language
CSS – Cascading Style Sheets
JS – JavaScript
FCP – First Contentful Paint
LCP – Largest Contentful Paint
TTI – Time to Interactive
SI – Speed Index
TBT – Total Blocking Time
FID – First Input Delay
CLS – Cumulative Layout Shift
TTFB – Time to First Byte
CTA – Call to Action

Příloha

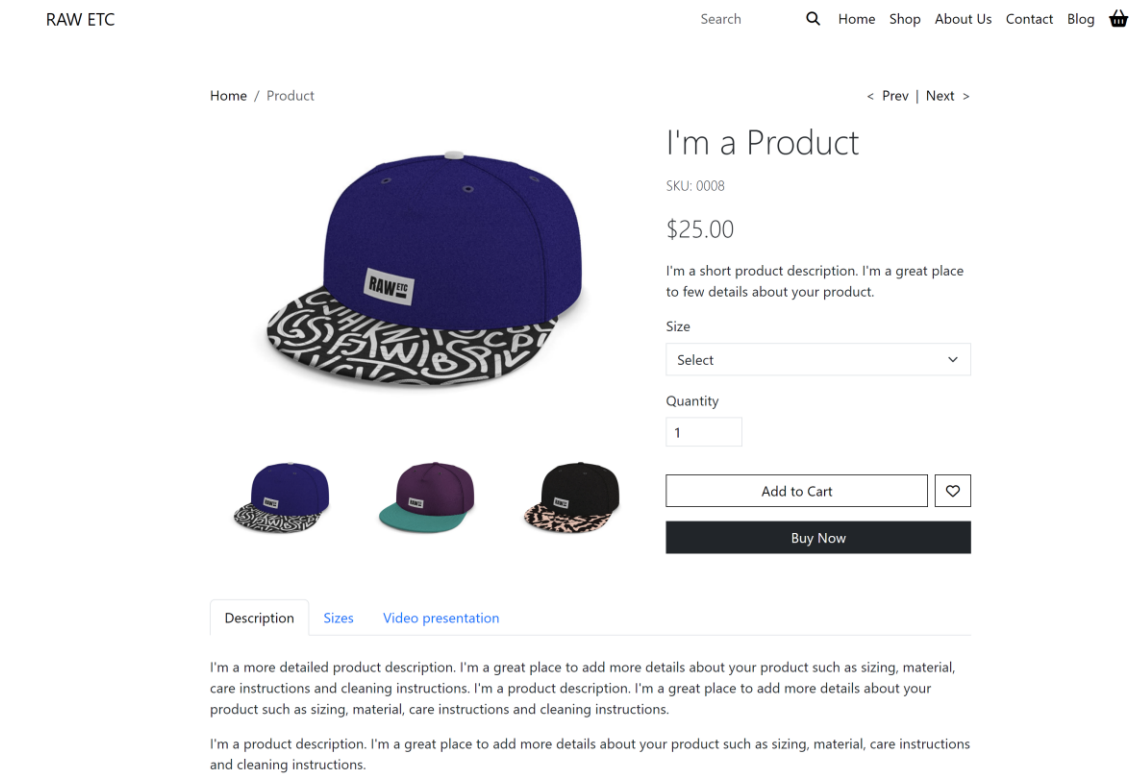
Příloha 1 – Zdrojová data uložena na přiloženém CD disku

Příloha 2 – Zdrojový kód uložen na přiloženém CD disku

Příloha 3 – Návrh úvodní stránky



Příloha 4 – Návrh stránky s detailem produktu



Příloha 5 – Návrh stránky s představením společnosti