Jihočeská univerzita v Českých Budějovicích

Přírodovědecká fakulta

Optimalizace webového frontendu

Diplomová práce

Bc. Lukáš Pavel

Vedoucí práce: Ing. Jiří Jelínek, CSc.

České Budějovice 2019

Bibliografické údaje

Pavel L., 2019: Optimalizace webového frontendu. [Web frontend optimization. Mgr.. Thesis, in Czech.] – 101 p. (počet stran), Faculty of Science, The University of South Bohemia, České Budějovice, Czech Republic.

Anotace

Práce se zabývá optimalizací rychlosti načítání webového frontendu a návrhem vlastní metodiky pro jeho optimalizaci a měření. Obsahem první části je přehled technologií používaných v současnosti pro návrh a vývoj na webovém frontendu. Dále je zpracována rešerše na téma metod optimalizace a metrik využívaných pro měření. Následně je vytvořena vlastní metodika na optimalizaci frontendové části webové prezentace. V praktické části je tato metodika ověřována na webové prezentaci firmy ČR Beton Bohemia spol. s r.o. včetně jejího vyhodnocení.

Annotation

This diploma thesis aims at optimalization of web frontend speed performance and creation of own methodolology for optimalization and measurment. The first part contains an overview of technologies currently used for design and development on a web frontend. The second part is research of optimization methods and metrics used for measurement. There is created my own methodolology for optimizing the web frontend in the third part of the thesis. The practical part consist of verifying the methodology based on the web presentation of the company ČR Beton Bohemia spol. s r.o. It also includes its evaluation.

Prohlašuji, že svoji diplomovou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své diplomové práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V ..... dne ..... Podpis autora

Poděkování

Rád bych poděkoval panu Ing. Jiřímu Jelínkovi, CSc., nejen za to, že mi umožnil zabývat se tímto zajímavým tématem, ale hlavně za odborné rady a celkovou korekturu práce. Děkuji také svým přátelům, rodině a přítelkyni za velikou podporu při tvorbě práce. V neposlední řadě patří poděkování také firmě ČR Beton Bohemia spol. s r.o. za to, že mi umožnila provést praktickou část diplomové práce.

Obsah

Úvod **8**

Cíle **10**

1 Technologie na webovém frontendu **11**

1.1 HTML . . . 11

1.2 CSS . . . 11

1.2.1 CSS frameworky . . . 12

1.2.2 CSS preprocesory . . . 13

1.3 JavaScript . . . 14

1.3.1 Angular . . . 15

1.3.2 React . . . 15

1.3.3 Vue.js . . . 15

1.3.4 Porovnání Angular, React a Vue.js . . . 16

1.4 Buildovací nástroje . . . 16

1.4.1 Webpack . . . 17

1.4.2 Gulp a Grunt . . . 17

2 Metriky **19**

2.1 Document.readyState . . . 20

2.2 Metriky v prohlížeči . . . 21

2.3 Obecné metriky . . . 22

2.4 Vlastní metriky . . . 23

2.5 Výběr metrik . . . 24

3 Metody optimalizace rychlosti načítání webového frontendu **25**

3.1 Technologie . . . 25

3.1.1 HTTP . . . 25

3.1.2 HTTPs . . . 26

3.1.3 HTTP/2 . . . 27

3.1.4 HTTP/3 . . . 28
3.1.5 HTTP komprese . . . 29
3.2 Objem dat . . . 31
3.2.1 Formáty obrázků . . . 31
3.2.2 Způsob zobrazení obrázků . . . 35
3.2.3 Fonty . . . 36
3.2.4 Minifikace HTML, CSS, JavaScript . . . 40
3.3 Čas . . . 40
3.3.1 LazyLoading . . . 40
3.3.2 Asynchronous vs Deferred JavaScript . . . 41
3.3.3 Kritické CSS . . . 41
3.3.4 Kritický Javascript . . . 42
3.3.5 Přednačítání - prefetching . . . 42
3.4 Server a zdroje . . . 44
3.4.1 Cache . . . 45
3.4.2 CDN . . . 46
3.5 Měření rychlosti webové aplikace . . . 47
3.5.1 Metodika RAIL . . . 47
3.5.2 Nástroje pro měření rychlosti . . . 49

4 Metodika optimalizace webové prezentace 53
4.1 Poznání firmy a jejich současné webové prezentace . . . 54
4.1.1 Analýza webové prezentace . . . 54
4.2 Definice obecných metrik . . . 56
4.3 Definice vlastních metrik . . . 57
4.4 Definice způsobu měření . . . 59
4.5 Zaznamenání hodnot před optimalizací . . . 59
4.6 Optimalizace . . . 60
4.6.1 Optimalizace technologií . . . 60
4.6.2 Optimalizace objemu . . . 61
4.6.3 Optimalizace času . . . 62

4.6.4 Optimalizace serveru a zdrojů 63
4.7 Zaznamenání hodnot po optimalizaci 64
4.8 Vyhodnocení optimalizace 64
4.9 Nastavení procesu pro pravidelné sledování rychlosti načítání . . . 64

5 Optimalizace webové prezentace pro firmu ČR Beton Bohemia spol. s r.o. 65

5.1 Poznání firmy a jejich současné webové prezentace 65
5.1.1 Analýza webové prezentace 66
5.2 Definice obecných metrik 70
5.3 Definice vlastních metrik 71
5.4 Definice způsobu měření 71
5.5 Zaznamenání hodnot před optimalizací 72
5.6 Optimalizace 73
5.6.1 Optimalizace technologií 73
5.6.2 Optimalizace objemu dat 75
5.6.3 Optimalizace času 78
5.6.4 Optimalizace serveru a zdrojů 81
5.7 Zaznamenání hodnot po optimalizaci 83
5.8 Vyhodnocení optimalizace 85
5.9 Nastavení procesu pro pravidelné sledování rychlosti načítání . . . 87

Závěr 88

Seznam použité literatury 90

Seznam tabulek 99

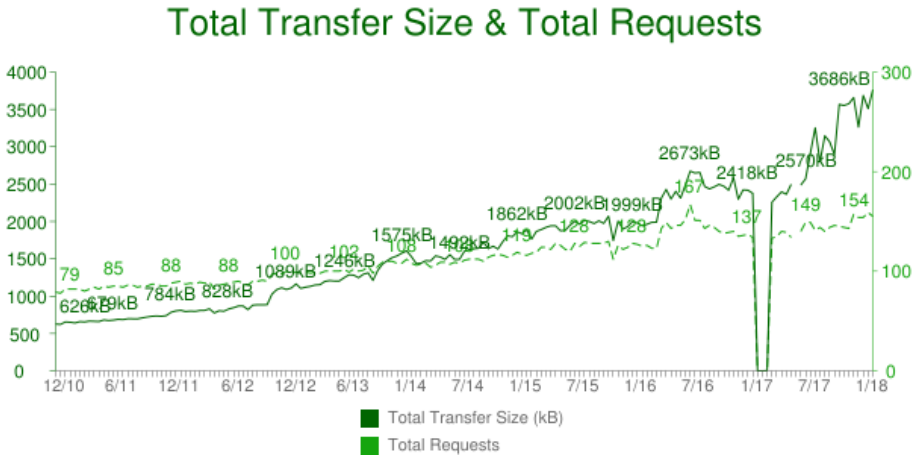
Seznam použitých zkratek 100

Úvod

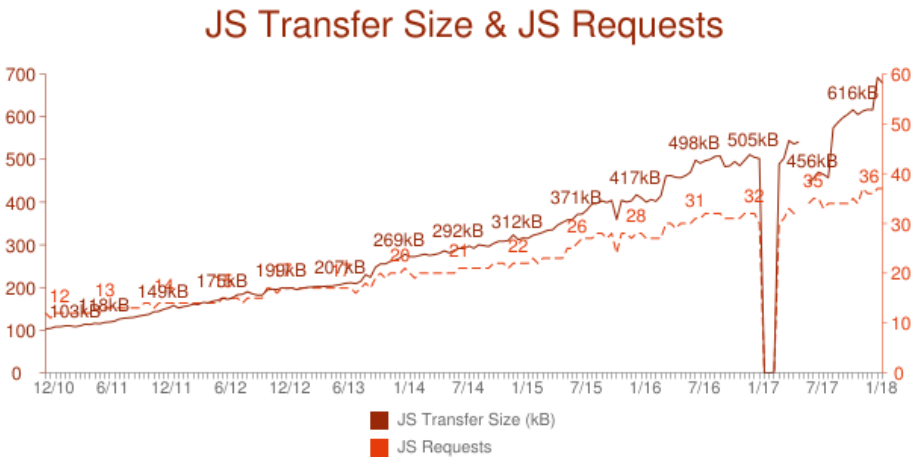
Od 6. srpna 1991, kdy CERN vypustil do světa první webovou aplikaci, se metody pro vývoj a návrh webové aplikace prakticky neustále vyvíjejí a zdokonalují. Ať už jde o frontendovou část aplikace, neboli část, kterou uživatel přímo vidí, anebo backendovou část, která řeší logiku aplikace.

Jeden z důležitých parametrů aplikace je i její rychlost. Pokud vyvineme aplikaci, která má skvělý design, je přínosná pro uživatele, ale je nepoužitelná kvůli své rychlosti, tak v dnešním konkurenčním světě selže a nemá téměř šanci na úspěch.

Před deseti lety byla rychlost načítání nejvíce zpomalována typem a rychlostí internetového připojení. Dnes je tomu ale úplně jinak. V dnešní době je načítání zpomalováno v první řadě rychlostí procesoru, což je způsobeno nárůstem velikosti JavaScript souborů, a v druhé řadě zařízeními, kde jsou aplikace využívány (mobilní zařízení) [10]. Za posledních osm let vzrostl počet stahovaných scriptů z 12 na 36 a jejich velikost z 103kB na 616kB na top 1000 webech na světě podle HTTP Archive viz obrázek 1. Nárůst můžeme také sledovat na celkové velikosti stránky a jejich požadavků na server, viz obrázek 2 [12].



Obrázek 1: Velikost přenášených dat podle HTTP Archive [12]



Obrázek 2: Velikost přenášeného JavaScriptu podle HTTP Archive [12]

Cíle

- Vytvořit přehled technologií používaných v současnosti pro návrh webového frontendu.
- Provést komentovanou rešerši na téma metod optimalizace webového frontendu podle různých hledisek (technologie, objem dat, čas, zdroje, atd.).
- Rozvinout metriky pro měření dopadu optimalizačních opatření. Navrhnout vlastní metodiku pro jejich měření včetně návrhu vhodné testovací aplikace.
- S navrženou aplikací provést sérii experimentů zaměřených na ověření optimalizačních postupů z předchozího bodu.

1. Technologie na webovém frontendu

1.1 HTML

HyperText Markup Language je značkovací jazyk pro vytváření webových aplikací v systému World Wide Web, který umožňuje publikaci dokumentů na Internetu [4]. Jazyk vychází z jazyka SGML a byl navržen roku v 90tých letech minulého stoletím Timem Berners-Leem. Následoval rychlý vývoj Internetu, a proto bylo nutné definovat pro HTML standardy. V lednu 1997 ve verzi 3.2 přebrala odpovědnost za tvoření standardů komunita W3C [4]. Na začátku roku 2000 byla vydána specifikace jazyka XHTML 1.0, která syntaxí vycházela z XML, nikoli ze SGML jako HTML. Počet dostupných značek a jejich význam zůstal stejný, jen se drobně změnila syntaxe. Postupem času se ukázalo, že XHTML je slepá cesta.Proto se v roce 2007 se ve W3C plně zaměřili na vývoj standardu HTML5 [5]. Aktuální verze jazyka HTML je 5.2 a verze 5.3 už je ve fázi draftu [6].

1.2 CSS

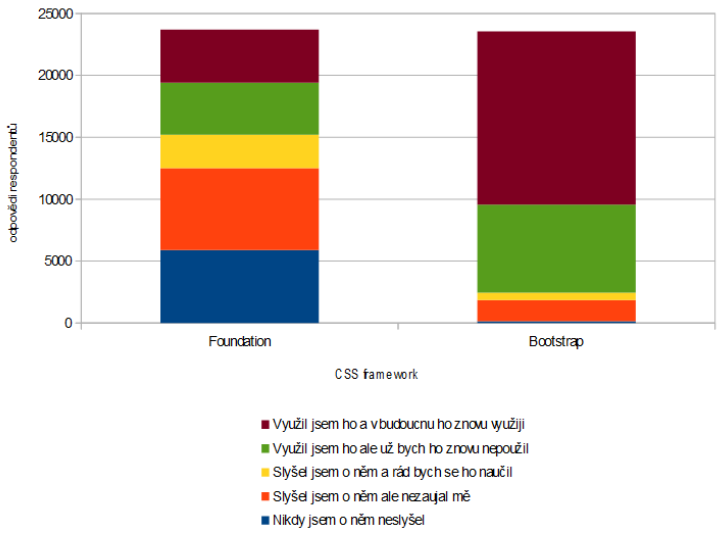
Kaskádové styly neboli jazyk CSS (Cascading Style Sheets) je jazyk, který popisuje zobrazení elementů na webu napsaných v jazycích HTML, XHTML nebo XML. Vývoj jazyka má na starost stejně jako HTML organizace WC3. Hlavní myšlenkou vývoje jazyka je oddělení obsahu webové stránky od jejího zobrazení. HTML má na starosti popis struktury a obsahu stránky, zatímco CSS se stará o její vzhled [7]. Když byla vydána první specifikace, byla prezentována jako jeden ucelený dokument CSS Level 1. Stejně tomu bylo i ve druhé specifikaci CSS Level 2. Od CSS Level 3 je ale specifikace rozdělena do jednotlivých modulů a každý modul se nezávisle dostává do těchto stupňů vývoje [8]:

- Working Draft (WD),

- Candidate Recommendation (CR),
- Recommendation (REC).

1.2.1 CSS frameworky

Jde o sadu nástrojů, které zefektivňují práci na webovém frontendu s typografií, tvorbou layoutu, vytváření elementů uživatelského rozhraní a opravují některé nedostatky napříč všemi platformami. Obsahují předdefinované komponenty, které lze použít při vývoji aplikace. Zpravidla jde například o modální okna, navigace, dropdowny, tlačítka nebo karusely [1]. Jejich využití ale nemusí být vždy výhodou. CSS frameworky nejsou vhodné pro webové aplikace s grafikou na míru z důvodu ohýbání předpřipravených stylů. Naopak jsou vhodné pro rychlý vývoj prototypu k testování nebo například pro jednoduché webové aplikace, kde není primární unikátní vzhled. Nejčastěji se využívají komponenty pro tvorbu layoutu, modálních oken a tlačítek. Podle výsledků ankety zveřejněných na webu stateofjs.com [2], kterou tvoří Raphaël Benitte, Sacha Greif a Michael Rambeau, je nejvíce využívaným CSS frameworkem Bootstrap, následuje Foundation, viz obrázek 1.1.



Obrázek 1.1: Využívání frameworků Bootstrap a Foundation podle stateofjs.com [2]

1.2.2 CSS preprocesory

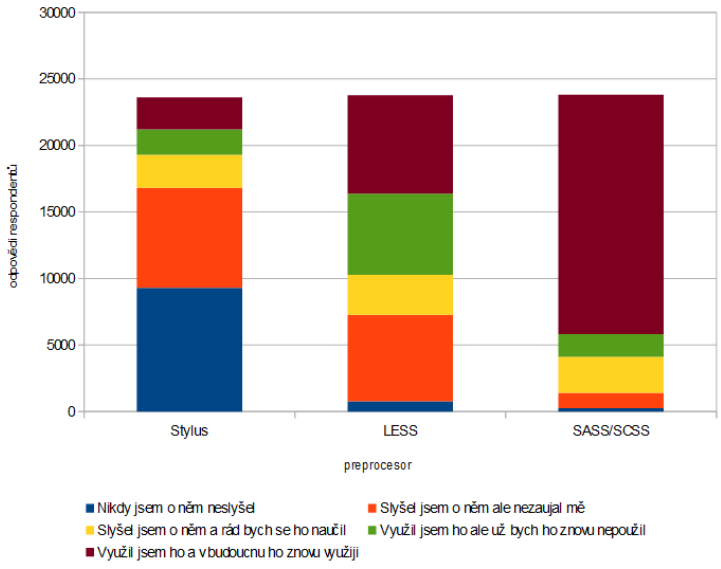
CSS preprocesor je skriptovací jazyk, který rozšiřuje jazyk CSS a je následně kompilován do CSS. Tyto preprocesory se využívají k tomu, aby kód byl více přehlednější, znovupoužitelný, udržovatelný a mohl obsahovat jednoduchou logiku [3]. Nejznámějšími preprocesory jsou:

- SASS,
- LESS,
- Stylus.

Mezi nejvíce využívanými vlastnostmi preprocesorů jsou:

- proměnné,
- mixiny,
- zanořování,
- matematické operace.

Podle webu stateofjs.com je v současnosti nejvíce využívaným preprocesorem SASS, viz obrázek 1.2.



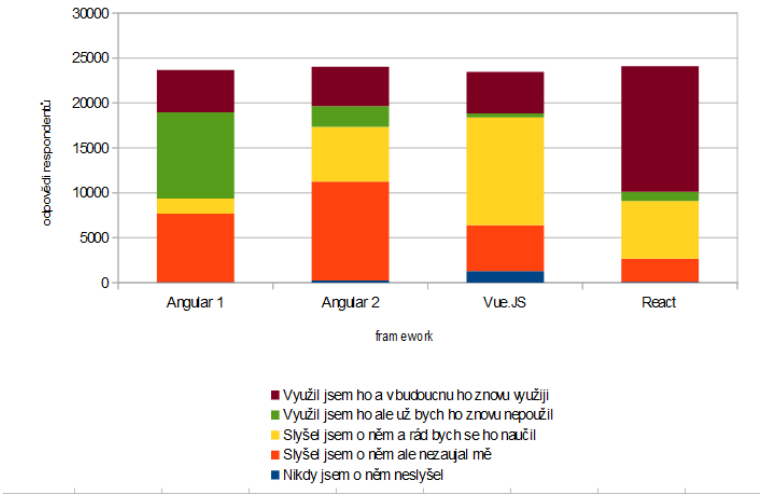
Obrázek 1.2: Využití CSS preprocesorů podle stateofjs.com [2]

1.3 JavaScript

JavaScript je interpretovaný, objektově orientovaný jazyk. Je znám zejména jako multiplatformní skriptovací jazyk pro webové aplikace. Nejdříve sloužil především pro manipulaci se strukturou DOM a úpravu vlastností HTML tagů. S vývojem webových technologií se ale postupně rozšířil a dnes najdeme JavaScript i na webovém serveru (Node.js). Tento prototypový, multiparadigmatický skriptovací jazyk je dynamický a podporuje jak objektově orientovaný, tak i imperativní a funkcionální styl programování.

Standardem pro JavaScript je ECMAScript, který má na starost ECMA International. Všechny moderní prohlížeče od roku 2012 podporují ECMAScript 5.1. Starší prohlížeče podporují maximálně ECMAScript 3. V červnu 2015 zveřejnila ECMA International šestou majoritní verzi označovanou jako ECMAScript 6 nebo také ES6. Od této verze se pravidelně vydávají nové verze v ročních cyklech [9].

V posledních letech se k tvorbě frontendové části aplikace využívají javascriptové frameworky. Ty nejrozšířenější a nejznámější mezi vývojáři jsou podle webu stateofjs.com React, Angular a Vue, jak ukazuje obrázek 1.3. Z grafu je patrné, že více než polovina vývojářů, kteří použili React, by ho použila znovu. Také je velice dobře zřetelný nárůst zájmu o Vue jakožto nejmladšího frameworku.



Obrázek 1.3: Nejrozšířenější javascriptové frameworky

1.3.1 Angular

Angular je javascriptový webový framework vyvíjený a udržovaný firmou Google. Angular (nebo také “Angular 2+”, “Angular 2” nebo ”ng2”) je následovník AngularJS (nebo také “Angular.js” nebo také “Angular 1.x”), ve kterém došlo k velkým změnám a je zpětně nekompatibilní s první verzí. Migrace z Angular JS na Angular 2 často znamená celou aplikaci přepsat. Angular 2 byl představen v září roku 2016 a je chápán jako reakce Googlu na React, kterým se i inspiroval. Verze 3 byla přeskočena a pokračovalo se další major verzí 4. Aktuální verze Angularu je 5.1.3.

Nejčastěji uváděnou výhodou je použití TypeScriptu jako vývojového jazyka. Ten je na rozdíl od JavaScriptu silně typový. To zapříčinilo snadný přechod vývojářů z tradičních objektově orientovaných jazyků jako je Java nebo C#, jelikož TypeScript se těmito jazyky silně inspiruje.

1.3.2 React

React je popisován jako "Javascriptová knihovna pro vytvoření uživatelských rozhraní”. Poprvé byl představen v březnu 2013. React je vyvíjen a udržován firmou Facebook. React využívá virtuální DOM k tomu, aby nebylo potřebné vykreslovat celý reálný DOM při každé změně. Změny v reálném DOM jsou totiž velmi pomalé. Při změně stavu se rychle provede porovnání a zvolí se nejefektivnější cesta, jak změnit reálný DOM za co nejmenší cenu, což znamená, že se překreslí pouze daná komponenta. Aktuální verze je 16.2.0.

1.3.3 Vue.js

Vue.js vyvinul a publikoval v únoru 2014 Evan You poté, co pracoval pro Google a vyvíjel jeho framework AngularJS. Pro svůj projekt potřeboval něco flexibilního a nenáročného pro čisté prototypování uživatelského rozhraní. Nenašel ale nic, s čím by se spokojil, a tak vznikl Vue.js. V začátku se tedy jednalo o velice efektivní nástroj pro rychlé prototypování. Nyní je však využíván k vývoji komplexních webových aplikací. Aktuálně nabízí jednoduchou cestu k využívání

a vytváření pluginů, psaní a využívání mixinů a celkovému psaní logiky aplikace. Vue.js se těší velké oblibě a zájem o tento framework stále narůstá. Aktuálně je ve verzi 2.5.13.

1.3.4 Porovnání Angular, React a Vue.js

Jak uvádí Shaumik Daityari ve svém článku [65]. Každý z těchto frameworků má své výhody a nevýhody. Výběr frameworku tak záleží na konkrétním typu projektu a jeho požadavcích pro vývoj. Dále uvádí základní porovnání, viz tabulka níže.

	Angular	React	Vue.js
Vznik	2010	2013	2014
Oficiální stránka	angular.io	reactjs.org	vuejs.org
Velikost (KB)	500	100	80
Framework využívá	Google, Wix	Facebook, Uber	Alibaba, GitLab
Počet sledujících na GitHubu	3300	37000	57000
Počet hvězd na GitHubu	43000	71000	122000
Počet stažení	11000	16000	17000
Commity za poslední měsíc	446	339	81
Počet vývojářů, kteří se podílí na vývoji	798	18000	240

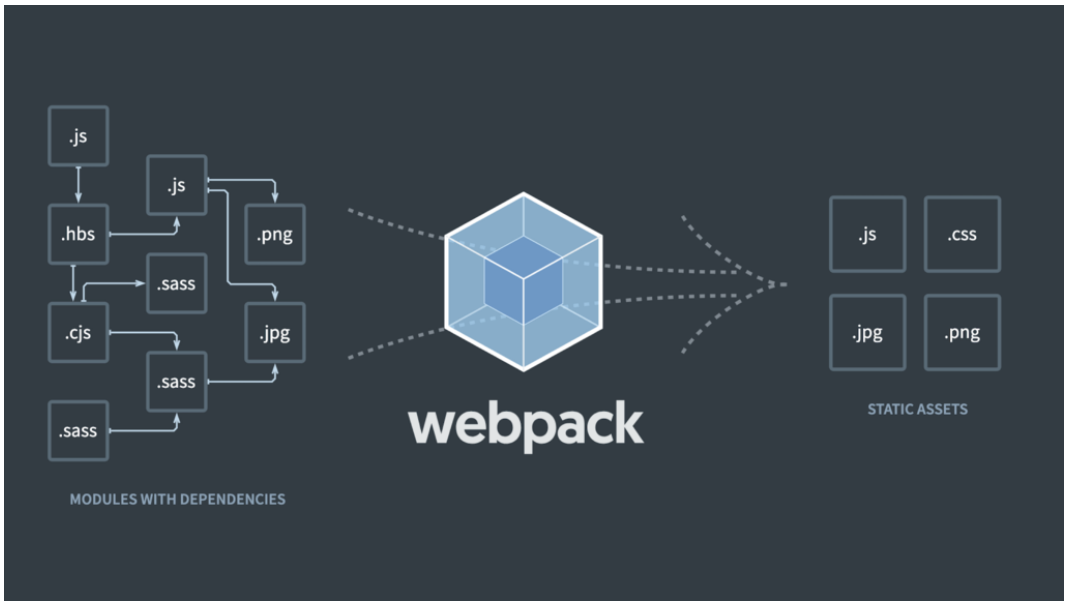
Tabulka 1.1: Základní porovnání Angular, React a Vue.js podle Shaumika Daityari [65]

1.4 Buildovací nástroje

Jedná se o automatizační nástroje, které jsou schopné zjednodušit a urychlit vývoj aplikace. Všechny tři zmíněné nástroje pro svůj běh využívají Node.js, což je run-time prostředí, které obsahuje vše, co potřebujeme ke spuštění programu napsaného v jazyce JavaScript [35].

1.4.1 Webpack

Webpack je nástroj, kterému za pomoci konfiguračního souboru můžeme určit jakým způsobem spojovat, umisťovat nebo upravovat zdrojový kód a soubory. Webpack při spuštění projektu prochází importy, vytváří graf závislostí projektu a poté generuje výstup založený na jeho konfiguraci. V konfiguraci popisujeme například jak načíst javascriptové soubory nebo jakým pohledem se dívat na SASS styly. Pomocí webpacku lze také vytvořit vývojové prostředí. Webpack se dále využívá pro výsledný export projektu do produkčního prostředí. Při spuštění projde projekt a podle konfigurace vytvoří tzv. bundles, což jsou výsledné optimalizované scripty a další soubory.



Obrázek 1.4: Princip webpack [63]

1.4.2 Gulp a Grunt

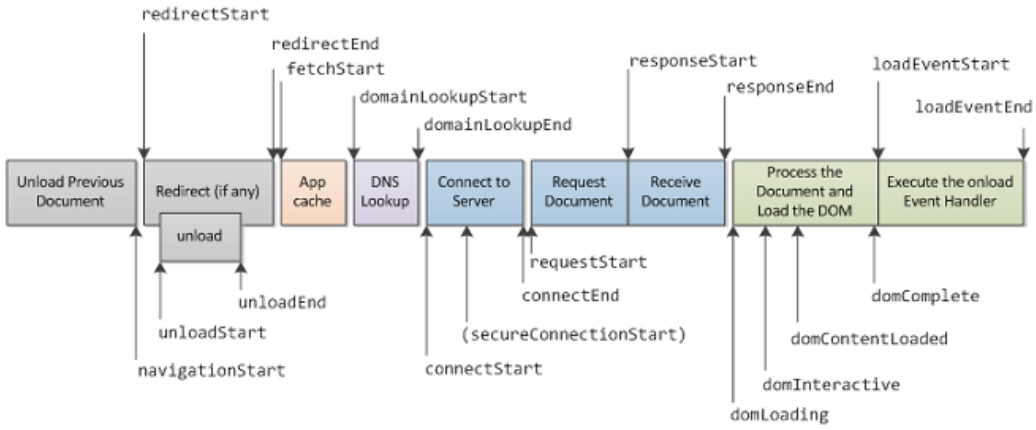
Gulp i Grunt jsou nástroje, které jsou schopny automatizovat opakující se úlohy, které by jinak dělal vývojář manuálně. Základní akce, které lze pomocí automatizačních nástrojů Grunt JS a Gulp využívat jsou:

- kompilace,
- minifikace souborů,

- spojování souborů,
- vytváření složek a souborů,
- kontrola syntaxe,
- tvorba dokumentace.

2. Metriky

Většina metrik měřených v rámci rychlosti načítání webové stránky je měřitelný čas, který uplyne od doby, kdy vznikne požadavek na webovou stránku, až po určitou událost v prohlížeči při načítání nebo vykreslování webové stránky, jak ukazuje obrázek níže.



Obrázek 2.1: Časová posloupnost událostí při načítání stránky [68]

Způsoby měření rychlosti načítání můžeme rozdělit na dvě základní skupiny, a to podle toho, jakým způsobem daná data získáváme. Jedná se o:

- Syntetická měření - měření, kdy data generuje stroj. Na aplikaci pošleme robota, který simuluje reálného uživatele. U většiny nástrojů lze konfigurovat, o jaký prohlížeč se jedná, jeho rozlišení a rychlost připojení k internetu.
- Měření reálných uživatelů (RUM - real user monitoring) - měření, kdy data posílá reálný uživatel aplikace. Většinou se využívá scriptů vložených do stránky. Velkou nevýhodou těchto měření jsou časové nároky na měření a náklady s tím spojené. Nicméně lze dnes taková data získat u větších webů na Page Speed Insights a Google postupně generuje data webů menších.

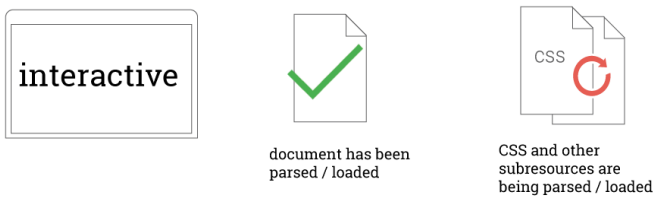
2.1 Document.readyState

Jedná se o stav dokumentu, který jednoznačně identifikuje jeho stav načtení v prohlížeči. Je používaný vývojáři k interakci se stránkou v určitém bodu načítání a také využívaný jako reference na událost nástroji pro optimalizaci a API rozhraními. Jeho stav je definován prohlížečem. Document.readyState má tři možné hodnoty [16]:

- Loading - dokument (např. HTML soubor) začíná být zpracováván a nahráván. V tomto bodě se uživateli zobrazuje prázdná stránka, neboť nejsou načteny žádné zdroje.

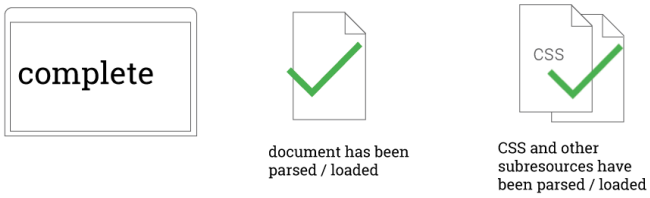
Obrázek 2.2: Stav Document.readyState = loading [16]

- Interactive - dokument je zpracován a nahrán, ale ostatní zdroje, jako je například CSS nebo obrázky, začínají být teprve nahrávány a zpracovávány. Uživateli se stále zobrazuje prázdná stránka.



Obrázek 2.3: Stav Document.readyState = interactive [16]

- Complete - dokument je zpracován, nahrán a stejně tak všechny známé další zdroje. Uživateli se začíná zobrazovat stránka.



Obrázek 2.4: Stav Document.readyState = complete [16]

2.2 Metriky v prohlížeči

Jedná se o stavy načtení HTML stránky, které lze měřit pomocí JavaScriptu:

- Time to first byte (TTFB) - čas od chvíle, kdy uživatel nebo klient vytvoří HTTP požadavek a první byte stránky dorazí do klientského prohlížeče. TTFB se skládá ze součtu tří časů. Čas, za který se požadavek zpropaguje přes síť k webovému serveru. Čas, za který server zpracuje požadavek a vygeneruje odpověď. Čas, za který se odpověď zpropaguje zpět přes síť do prohlížeče [15].

- DOM Loading - je čas, kdy prohlížeč obdržel HTML dokument a je připraven s ním dále pracovat. Stav dokumentu dostává hodnotu "loading", to znamená, že dokument (např. HTML soubor) začíná být zpracováván [17].

- DOM Interactive - okamžik, kdy prohlížeč dokončil zpracování HTML a sestavil DOM. Stav dokumentu dostává hodnotu "interactive". To znamená, že začínají být zpracovávány ostatní zdroje [18].

- DOM ContentLoaded - moment, kdy DOM a CSSOM je načten a neexistují styly, které by bránily ve zpracování JavaScriptu [19].

- DOM Complete - je čas, kdy je zpracované HTML a konstrukce DOM je kompletní a ostatní zdroje jsou také staženy. Stav dokumentu dostává hodnotu "complete". Jde o stav, kdy prohlížeč přestává zobrazovat známé

běžící kolečko znázorňující načítání stránky. Pro vývojáře je to čas k načítání další aplikační logiky nebo dalšího JavaScriptu.

- Load Event - okamžik, kdy je objekt načten. Využívá se často s elementem <body> ke spuštění scriptu, když je stránka načtena.

2.3 Obecné metriky

Nyní si ukážeme základní metriky, které představil Steve Souders [10]:

- Speed Index - je průměrný čas zobrazení konkrétní stránky v daném prohlížeči, velikosti okna a rychlosti internetu, neboli ukazuje, jak rychle je viditelný obsah stránky naplněn do stavu stoprocentního vykreslení. Čím nižší je náš Speed Index, tím lépe. Nejrychlejší weby dosahují čísel kolem tisícovky. Průměr je mezi pěti a deseti tisíci. Čísla nad deset tisíc jsou na pováženou [11]. Níže můžeme vidět vzorec pro výpočet Speed Indexu [16].

$$SpeedIndex = \int_0^{end} 1 - \frac{vc}{100}$$

end - end time in miliseconds

vc - % visually complete

- Start Render - je čas, kdy je vykreslen první viditelný obsah na stránce, ať už je jakýkoliv.

- First Meaningful Paint - okamžik, kdy bylo vykresleno na stránce něco smysluplného.

- Time to First Interactive - moment, kdy se na stránce očekává, že bude použitelná a bude reagovat na vstup. Konkrétně se jedná o první rozpětí 5 sekund, kdy hlavní vlákno prohlížeče nikdy nebylo zablokováno po více než 0 ms po First Meaningful Paint.

- Time to Consistently Interactive (neboli Time to Interactive) - je stav, kdy je stránka použitelná a reaguje konzistentně na vstup. Stejně jako u Time to First Interactive, ale nejsou na cestě více jak dva požadavky nebo odpovědi na server.
- Visually Complete - moment, kdy je stránka kompletně načtena.

Tyto metriky lze měřit pomocí většiny nástrojů pro měření rychlosti webové prezentace jako jsou například WebPageTest, Chrome Dev Tools, Lighthouse a další.

2.4 Vlastní metriky

Neměli bychom se však omezovat pouze na obecné metriky. Lze si definovat i své vlastní metriky, které jsou pro danou aplikaci důležité. Velmi často se tyto metriky překrývají s cílem aplikace, respektive jejich jednotlivých stránek a stavů. Například pokud provozujeme eshop a zaměříme se na stránku detailu produktu. Na takové stránce nás nezajímá, kdy bude plně načtena, ale zásadní pro nás je, za jaký čas se uživateli zobrazí tlačítko koupit, případně, kdy se načte první obrázek produktu, jak uvádí ve své přednášce Dominik Tilp na konferenci Barcamp Brno 2017 [14]. Další vlastní metriky, které pan Tilp uvádí a patří mezi nejčastěji definované jsou:

- uživatel je schopen přečíst článek,
- uživatel je schopen číst komentáře,
- uživatel je schopen vidět dostupnost produktu.

Vlastní metriky, které jsou definované jako zobrazení elementu uživateli, jako jsou například výše zmíněné od pana Tilpa, lze měřit pomocí Element Timing API od WICG (The Web Platform Incubator Community Group) [70].

2.5 Výběr metrik

Výběr metrik, které budeme sledovat je velice důležitý a závislý na cíli celé aplikace nebo stránky. Ve většině případů nás nejvíce zajímají právě vlastní definované metriky. Následují obecné metriky, neboť vyhledávače používají rychlost jako jeden z hodnotících faktorů pro zobrazení ve fulltextu. Jako poslední jsou časy událostí v prohlížeči. Podle sestavení metrik také přistupujeme k optimalizaci webové aplikace.

3. Metody optimalizace rychlosti načítání webového frontendu

3.1 Technologie

3.1.1 HTTP

HTTP (Hypertext Transfer Protocol) je bezstavový internetový protokol na aplikační vrstvě podle modelu ISO/OSI, určený pro výměnu hypertextových dokumentů ve formátu HTML. V současnosti je však používán i pro přenos dalších informací a typů souborů díky rozšíření MIME. Samotný protokol HTTP neumožňuje šifrování ani zabezpečení integrity dat. Protokol funguje na principu dotaz-odpověď a v dnešním světě internetu je používán především ke komunikaci prohlížeče se serverem. Používá obvykle port TCP/80. Jeho první zdokumentovaná verze 0.9 byla představena v roce 1991. Následovaly verze 1.0 v roce 1996, verze 1.1 v roce následujícím a jeho poslední a aktuální verze 2.0, která byla publikována v roce 2015 jako RFC 7540 [28].

Příklad požadavku na sever v HTTP1.1

```
1 GET /index.html HTTP/1.1
2 Host: www.priklad.cz
```

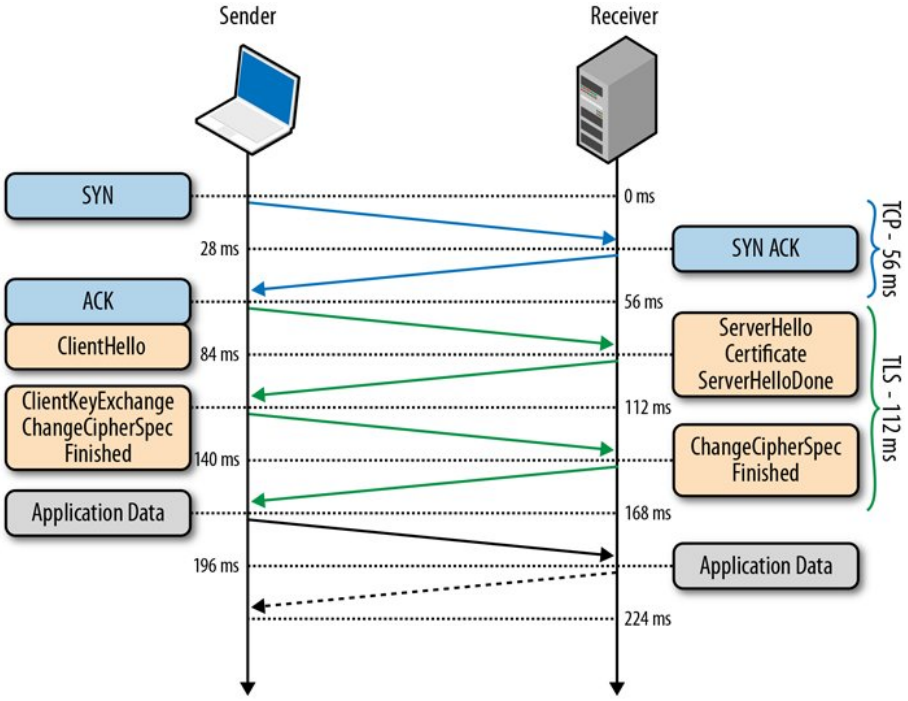
Příklad odpovědi ze serveru

```
1 HTTP/1.1 200 OK
2 Date: Mon, 23 May 2005 22:38:34 GMT
3 Content-Type: text/html; charset=UTF-8
4 Content-Encoding: UTF-8
5 Content-Length: 138
6 Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
7 Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
8 ETag: "3f80f-1b6-3e1cb03b"
9 Accept-Ranges: bytes
```

```
10 Connection: close
11
12 <html>
13 <head>
14   <title>An Example Page</title>
15 </head>
16 <body>
17   Hello World, this is~a very simple HTML document.
18 </body>
19 </html>
```

3.1.2 HTTPs

Protokol HTTPS (Hypertext Transfer Protocol Secure) zajišťuje autentizaci, důvěru a integritu dat. Protokol HTTP sám o sobě nepodporuje šifrování přenášených dat, k tomu slouží protokol HTTPS. HTTPS protokol využívá HTTP, ale je ještě rozšířen o protokol TLS (Transport Layer Security) nebo SSL (Secure Sockets Layer). Používá obvykle port TCP/443 [30]. Jeho první verze byla představena v roce 1994 společností Netscape Communications, která ho vyvíjela pro webový prohlížeč Netscape Navigator [29].



Obrázek 3.1: Úvodní handshake HTTPS [31]

3.1.3 HTTP/2

HTTP2 je druhá major verze protokolu HTTP. Jeho základ vychází z protokolu SPDY, který vyvíjel Google. Specifikace byla publikována v květnu roku 2015 a skládá se ze dvou částí. První je samotná specifikace Hypertext Transfer Protocol version 2 (RFC7540) a druhá HPACK - Header Compression for HTTP/2 (RFC7541). HTTP/2 se zaměřuje na výkonnost, snížení latence u přenosu a snižování zátěže na serveru a síti [47] [48].

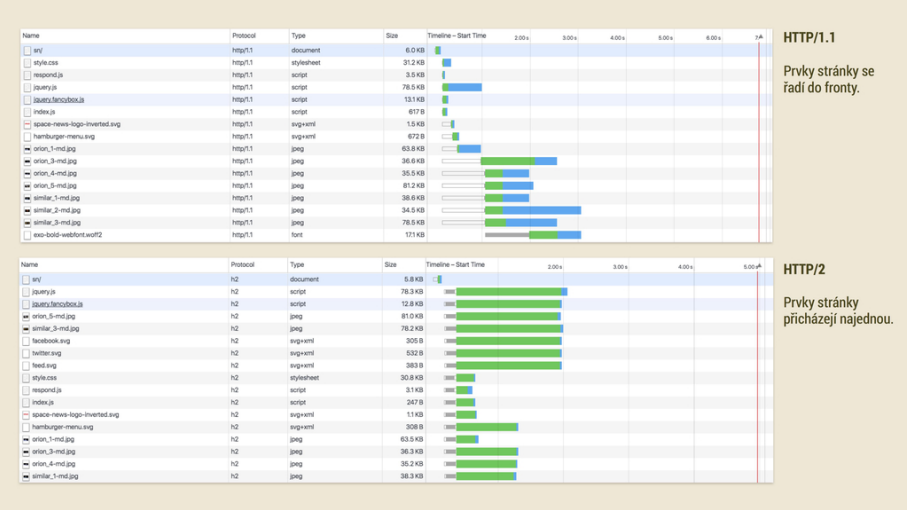
Výhody oproti HTTP popisují Dan Bernier a Martin Michálek takto [49] [50]:

- Protokol je binární, to znamená, že se rychleji parsuje a přenáší po síti.
- Podporuje komprimaci hlaviček za použití HPACK, který byl speciálně vyvinut pro protokol HTTP/2.
- Podporuje multiplexing - mezi prohlížečem a serverem je otevřeno jedno spojení, které využívá několik streamů, které jsou na sobě nezávislé.

- Podporuje Server push - už při prvním dotazu (na HTML) můžete server nastavit tak, aby poslal prvky stránky – vybraný obrázek, CSS nebo JS soubory.

- Podporuje priority - prohlížeče mohou například upřednostnit stahování CSS před obrázky. Využívá se zde priorit streamů.

Hlavní rozdíl oproti HTTP, který je důležitý pro optimalizaci frontendu, je především způsob přenosu ze serveru do prohlížeče. V HTTP se zdroje řadí do fronty, zatímco u HTTP/2 se zdroje stahují paralelně, jak ukazuje obrázek níže.



Obrázek 3.2: Rozdíl ve stahování zdrojů u HTTP a HTTP/2 [50]

Jonathan Griffin ve svém článku [64] uvádí rozdíl mezi HTTP a HTTP/2 z pohledu rychlosti načítání. Testoval sedm webových prezentací, u kterých sledoval metriku Document Complete. Ve výsledcích testování uvádí, že HTTP/2 je rychlejší v průměru až o 14%.

3.1.4 HTTP/3

Jak uvádí ve svém článku Jonathan Griffin [64] HTTP/3 je založen na protokolu QUIC (Quick UDP Internet Connections) od společnosti Google. Poprvé byl představen Markem Nottinghamem v říjnu roku 2018. Předpokládané vydání standardu HTTP/3 je očekáváno na konci roku 2019.

QUIC je podobný HTTTP/2, ale hlavní rozdíl je v tom, že je implementovaný na protokolu UDP namísto TCP. To má několik výhod a klíčových vlastností:

- pakety UDP jsou přijímány rychleji,
- odesílatel nečeká na potvrzení o doručení paketů,
- neprovádějí se žádné kontroly chyb,
- výrazně zkrácený čas navázání spojení,
- vylepšená kontrola přetížení,
- multiplexování bez blokování head-of-line,
- oprava chyby vpřed.

3.1.5 HTTP komprese

HTTP komprese snižuje dobu odezvy zmenšením velikosti odpovědi HTTP. Kompresi lze provádět dvěma způsoby [54]:

- Dynamicky - komprimuje soubory, když jej o to server v požadavku požádá, a je výchozím přístupem používaným většinou webových serverů. Dynamická komprimace je užitečná pro obsah, který se často mění, například webové stránky generované aplikací.
- Staticky - komprimuje každý soubor předem a dodává tuto předkomprimovanou verzi při požadavku na soubor. Soubory, které se často nemění (například JavaScript, CSS, písma a obrázky), jsou pro statickou kompresi nejvíce prospěšné, protože je třeba je komprimovat pouze jednou. Tím ušetří čas CPU serveru.

Gzip

Gzip je nejpopulárnější způsob komprese a obecně snižuje velikost odezvy přibližně o 70% [53]. Gzip může komprimovat téměř jakýkoliv typ souboru od prostého

textu po obrázky a je dostatečně rychlý k tomu, aby komprimoval a dekomprimoval data za běhu. Když webový server obdrží požadavek, vygeneruje odpověď a poté zkontroluje v hlavičce parametr Accept-Encoding a určí, jakým způsobem kódovat odpověď. Gzip nabízí rozsah úrovní komprese od 1 do 9 [54].

Brotli

V roce 2013 byl spuštěn nástroj Brotli pro offline kompresi webových fontů. Později, v roce 2015, společnost Google vydala verzi pro obecnou bezeztrátovou kompresi dat. Je to open source, překonává gzip a je podporován většinou prohlížečů [66]. Mike MacCana uvádí ve svém článku [67] že:

- JavaScript soubory kompresované pomocí Brotli byly 14% menší než při použití Gzip,
- HTML soubory byly 21% menší než při použití Gzip,
- CSS soubory byly 17% menší než při použití Gzip.

Jak Gzip, tak Brotli používají více úrovní komprese. Začínají od úrovně 1 a zvyšují se. Čím vyšší je úroveň komprese, tím menší je soubor, ale zároveň je komprese výpočetně náročnější a časově delší, jak ukazuje porovnání, které uvádí Paul Calvano ve svém článku [66].

Plugin	Codec	Level	Compression Ratio	Compression Speed (MB/s)	Decompression Speed (MB/s)
brotli	brotli	1	3.68	116.95	346.56
brotli	brotli	2	3.72	97.75	353.70
brotli	brotli	3	3.73	86.52	357.10
brotli	brotli	4	3.88	68.60	370.95
brotli	brotli	5	4.07	33.56	367.08
brotli	brotli	6	4.14	29.43	373.43
brotli	brotli	7	4.19	22.31	374.89
brotli	brotli	8	4.21	18.05	376.79
brotli	brotli	9	4.23	13.57	376.79
brotli	brotli	10	4.67	0.51	301.74
brotli	brotli	11	4.67	0.51	301.60
zlib	gzip	1	2.77	83.97	230.49
zlib	gzip	2	2.85	77.79	233.13
zlib	gzip	3	2.88	69.12	235.18
zlib	gzip	4	3.07	53.05	234.28
zlib	gzip	5	3.14	42.01	237.18
zlib	gzip	6	3.15	35.61	237.86
zlib	gzip	7	3.18	30.71	239.21
zlib	gzip	8	3.21	19.58	240.94
zlib	gzip	9	3.21	18.77	241.02

Obrázek 3.3: Porovnání Brotli a Gzip od Paula Calvaniho [66]

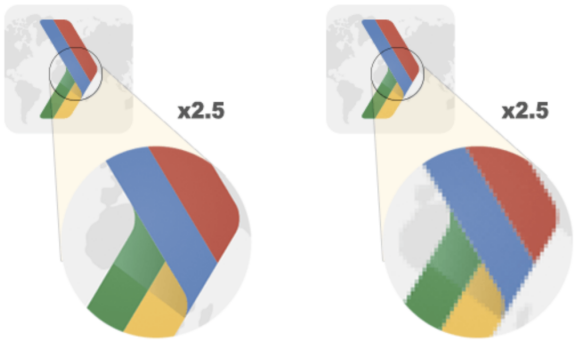
3.2 Objem dat

Obrázky jsou stále největší data, která stahujeme při procházení webu. Podle webu HTTP Archive, 60% dat stahovaných z webu představují obrázky ve formátu JPEG, PNG a GIF. Podle Tammy Everts zvyšuje přidání obrázků konverzní poměr stránek [21]. Protože obrázky na webu jsou a v budoucnu stále budou, je důležité se zaměřit na jejich optimalizaci. Nejsou to ale pouze obrázky, na které je třeba se zaměřit. Optimalizovat lze i ostatní soubory jako HTML, CSS nebo JavaScript.

3.2.1 Formáty obrázků

Jak Ilja Grigorik napsal ve svém článku, správný formát pro obrázek na webovou stránku je kombinací požadovaných vizuálních výsledků a funkčních požadavků [24]. V základu lze grafiku rozdělit na dvě hlavní skupiny [25]:

- Rastrová grafika - využívá k prezentaci obrázku jednotlivých pixelů (obrazových bodů). Obrázek je zanesen do mřížky, takzvaného rastru. Nevýhodou rastrového obrázku je ten, že při určitém zvětšení se viditelně zhorší jeho kvalita.
- Vektorová grafika - využívá křivek a bodů k zachycení obrazových informací. Obrázek vytvořený „ve vektorech“ se může libovolně zvětšovat či zmenšovat, aniž by vizuálně utrpěla jeho kvalita.



Obrázek 3.4: Rozdíl po zvětšení u rastrové a vektorové grafiky [24]

JPEG

Formát JPEG (Joint Photographic Experts Group) je pravděpodobně světově nejpoužívanější formát pro obrázky. Podle webu HTTP Archive je 45% obrázků na webu právě ve formátu JPEG. Jedná se o starší formát, který byl představen už v roce 1992.

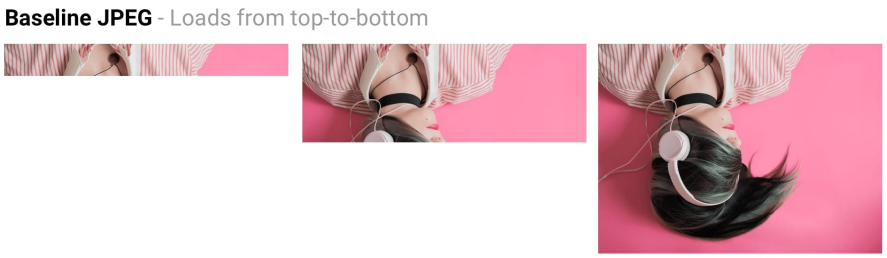
JPEG je představitelem rastrové grafiky a využívá ztrátový kompresní algoritmus, který je založen na diskrétní kosinové transformaci (DCT). Formát JPEG je vhodný především pro fotografie.

Většina optimalizačních nástrojů nám nabízí možnost určit si úroveň komprese obrázku. Vyšší komprese sice výrazně zmenší výslednou velikost, ale také může zanechat na obrázku artefakty, jak ukazuje ve svém průvodci Addy Osmani [25].

Obrázek 3.5: Různé úrovně komprese [24]

JPEG formát nabízí několik kompresních módů. Tři nejvíce využívané z nich jsou:

- Sekvenční JPEG (baseline) - je nejčastěji používaný. Kódování a dekódování probíhá směrem shora dolů. Sekvenční JPEG poznáme tak, že při pomalém připojení se načítá postupně od horního okraje.



Obrázek 3.6: Rozdíl po zvětšení u rastrové a vektorové grafiky [24]

- Progresivní JPEG (progresive, PJPEG) - progresivní JPEG rozdělí obrázek na několik vrstev. První vrstva zobrazuje obrázek rozmazaný a postupně s dalšími vrstvami se obrázek doostřuje.
- Bezztrátový JPEG (lossless) - bezztrátová optimalizace obrázku spočívá v odstranění EXIF dat, které přidávají fotoaparáty nebo editory. Optimalizuje se také huffmanova tabulka obrázku.

Obrázek 3.7: Rozdíl po zvětšení u rastrové a vektorové grafiky [24]

Postupné načítání obrázků, které nabízí PJPEG, vylepšuje vnímanou rychlost načítání pro uživatele. Velice často mu totiž k přečtení obsahu obrázku stačí i nižší kvalita. V roce 2015 facebook.com přepnul na PJPEG v jejich aplikaci pro iOS a pozoroval 10% snížení přenosu dat oproti sekvenčnímu JPEGu. Dále se až o 15% rychleji zobrazovaly obrázky v dobré kvalitě [26]. PJPEG využívá také twitter nebo pinterest.

PJPEG nemusí vždy být tou nejlepší volbou. Nevýhodou PJPEG je, že může být až 3x pomalejší dekódování oproti sekvenčnímu JPEG. To se může projevit především na mobilních zařízeních. Dále také u malých obrázků (například náhledy) může být PJPEG datová velikost vyšší než u sekvenčního nebo bezztrátového JPEGu.

PNG

PNG je zkratka pro Portable Network Graphics. Byl vyvinut v roce 1995 jako náhrada za formát GIF. Stejně jako GIF využívá bezztrátovou kompresi a nabízí 8-bitový nebo 24-bitový formát. Podporuje průhlednost, ale v základu formát PNG

nepodporuje animace. Až rozšíření MNG dovoluje využít animace, ale je zpětně nekompatibilní s PNG. Nicméně díky MNG vznikl formát APNG, který pokud není podporován, zobrazí statický PNG obrázek.

GIF

Graphics Interchange Format neboli GIF je formát určený pro rastrovou grafiku, který byl vyvinut společností CompuServe už v roce 1989. GIF používá bezztrátovou kompresi LZW. Formát GIF je 8-bitový a využívá barev z barevné palety (256 barev), což je jedna z nejvíce omezujících vlastností tohoto formátu. GIF umožňuje jednoduché animace [34].

WebP

WebP je moderní formát pro obrázky od společnosti Google, která ho představila v roce 2010. Poskytuje jak ztrátovou, tak bezztrátovou kompresi. Má možnost výrazného snížení velikosti výsledného obrázku oproti JPEG nebo PNG. Podporuje animace stejně jako GIF a podporuje také průhlednost alfa kanálu. Naopak nepodporuje progresivní načítání a subsampling chroma kanálu. Jeho zatím největším problémem je podpora v prohlížečích, kdy plnou podporu má zatím pouze v prohlížečích založených na Chrome. Ostatní prohlížeče jako Edge, Safari, nebo Firefox na podpoře teprve pracují nebo není úplná. Při využívání tohoto formátu musíme myslet i na záložní řešení pro prohlížeče bez podpory WebP [32].

Podle studie od společnosti Google dochází k zmenšení u formátu WebP oproti bezztrátových obrázků typu PNG o 26% a u ztrátových obrázků typu JPEG mezi 25% až 34% [33].

SVG

Scalable Vector Graphics neboli SVG není na rozdíl od ostatních rastrový formát, ale vektorový. Formát byl vyvinut v roce 2001 a využívá jazyk XML. Díky tomu je čitelný jak pro člověka, tak pro stroj. To znamená, že vyhledávače jsou schopny takový obrázek nejen přečíst, ale i v něm vyhledat například text. SVG využívá

matematicky deklarovaných tvarů a křivek, ne pixelů. Díky tomu je na jakémkoliv rozlišení a v jakékoliv velikosti vždy ostrý a současně je zachována datová velikost. SVG lze animovat, podporuje průhlednost a všechny kombinace barev nebo přechodů. Pokud generujeme SVG z grafického editoru, velice často obsahují metadata, informace o vrstvách a komentáře, které nejsou potřeba [24]. Proto je potřeba takové SVG optimalizovat, například pomocí nástroje SVGOMG [40].

3.2.2 Způsob zobrazení obrázků

Při práci s obrázky musíme myslet na to, že datovou náročnost souboru ovlivňuje nejen formát, ale i vlastní rozlišení obrázku. Na mobilních zařízeních ve většině případů nepotřebujeme tak velké rozlišení, jako na klasickém 15,6 palcovém monitoru. Naopak na displeji typu Retina (různými poměry device-pixel-ratio) potřebujeme větší rozlišení, aby byl obrázek ostrý. Je několik způsobů, jak zobrazovat obrázky v HTML. Nejsnadnější je to s formátem SVG. U SVG velikost nemusíme řešit, neboť jeho datová velikost je stále stejná a je ostrý v jakémkoliv rozlišení. U ostatních to lze řešit atributy srcset a sizes u elementu <img> nebo elementem <picture>, jak ukazuje Martin Michálek ve své příručce [37] [38].

Element <img srcset sizes>

Rozhodování, který obrázek se využije, necháváme čistě na prohlížeči. Ten se rozhoduje podle šířky okna prohlížeče a aktuální device-pixel-ratio. Pomocí atributu srcset mu definujeme více variant obrázku, které má na výběr. Atribut srcset obsahuje ještě dva descriptory vlastností obrázků. Prvním je descriptor w, který udává jakou pixelovou šířku má ve skutečnosti soubor s obrázkem. Druhým je x, který udává připravenost souboru s obrázkem pro různé poměry device-pixel-ratio. Atribut sizes udává velikost obrázku na dané stránce. V drtivé většině totiž nechceme, aby se velikost vybírala podle okna prohlížeče, ale podle rodičovského HTML elementu. Při využívání atributů srcset a sizes definujeme elementu i atribut src jako záložní řešení pro případ, že by prohlížeč tyto elementy nepodporoval. Na další straně následuje ukázka využití elementu <img> s atributy srcset a sizes od Martina Michálka [37]:

```
<img src="small.png"
  srcset="small_600.png 600w,  medium_1024.png 1024w,
     large_1600.png 1600w"
  sizes="(min-width: 768px) 300px, 100vw"
  width="200"
  height="200"
  alt="small"
>
```

Element <picture>

Rozhodování o tom, který obrázek se využije, udává vývojář. Respektive určuje, jaký zdroj preferovat. Prohlížeč pak vezme první vyhovující zdroj. Element <picture> obsahuje potomky <source>, které definují alternativy k poslednímu <img> elementu, jak ukazuje příkladvyužití elementu <picture> od Martina Michálka [38]:

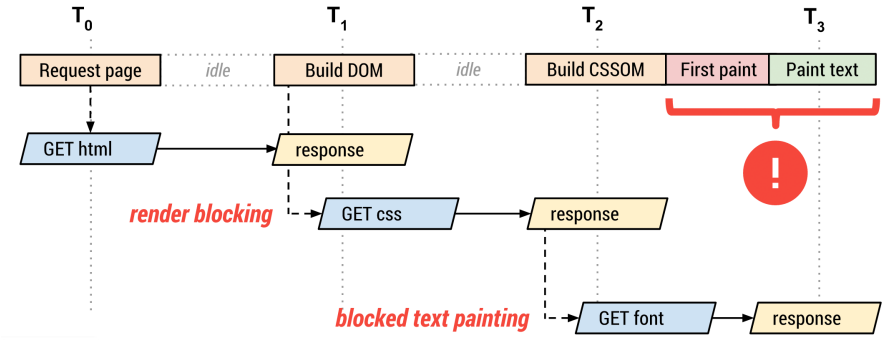
```
<picture>
  <source media="(min-width: 1024px)" srcset="large.jpg">
  <source media="(min-width: 600px)" srcset="medium.jpg">
  <img src="small.jpg" alt="small">
</picture>
```

Výhodou použití elementu <picture> je, že můžeme definovat více souborových formátů pro stejné rozlišení, dále pak kontrola nad tím, jaký obrázek se vybere. Častou nevýhodou jsou naopak velmi složitá pravidla a řazení jednotlivých variant, neboť element <picture> nerespektuje sám o sobě poměr device-pixel-ratio.

3.2.3 Fonty

Jak napsal Ilya Grigorik optimalizace webových fontů patří mezi kritické části v celkové optimalizaci načítání. Každý font představuje další datovou zátěž pro web a některé fonty mohou blokovat vykreslení textu. Nicméně optimalizované fonty, v kombinaci s rozumnou strategií pro jejich načtení a použití na stránce,

mohou pomoci snížit celkovou velikost stránky a zkrátit čas potřebný k vykreslení stránky. Základem při optimalizaci fontů je snížení jejich datové velikosti pomocí komprese. Další možností je řízení strategie načítání. Ve výchozím nastavení prohlížeče blokují vykreslení textu, dokud se nenačtou všechny fonty. Posledním základním blokem je nastavení správného cachování, neboť se často jedná o statický zdroj, který není často aktualizován [41].



Obrázek 3.8: Neošetřené nahrávání fontů [62]

Formáty webových fontů

Aktuálně se používají pro zobrazení textu čtyři formáty. Přesto však neexistuje formát, který by podporovaly všechny prohlížeče.

- EOT- podporují pouze Internet Explorer 8 a nižší verze.
- TTF - má podle webu caniuse.com pouze částečnou podporu v Internet Exploreru.
- WOFF - má největší podporu napříč moderními prohlížeči, avšak starší prohlížeče jako Internet Explorer 8 nebo Android 4.3 ho nepodporují.
- WOFF 2.0 - je nejnovější, ale aktuálně nemá podporu Internet Exploreru.

Optimalizace fontů v CSS

Úsporu můžeme vytvořit správným definováním vlastnosti @font-face:

- Parametr local() - nejdříve zjistí, zda již uživatel nemá nainstalován font na svém zařízení.

- Parametr format() - určujeme, o jaký formát se jedná a jeho prioritu.
- Parametr unicode-range - velké univerzální fonty můžeme rozdělit na menší, které obsahují pouze specifické symboly.
- Parametr font-style - ne vždy potřebujeme všechny styly daného fontu. Často například nepotřebujeme kurzívu.
- Parametr font-weight - stejně jako nepotřebujeme styly, tak nepotřebujeme všechny řezy fontu.
- Parametr font-display - definuje různé strategie pro načtení fontu. Zejména určuje dobu, po kterou blokuje vykreslení textu po požadavku na font. Může mít hodnoty auto, block, swap, fallback, optional.

Optimalizace fontů v HTML

Optimalizace spočívá ve využití elementu <link rel='preload'> v elementu <head> pro přednačtení požadováno fontu. Říkáme tak prohlížeči, aby při parsování HTML přednačetl tento soubor. On sice ještě neví, k čemu ho bude potřebovat, ale později už nečeká na sestavení CSSOM, aby vytvořil požadavek na stažení fontu. Tuto techniku lze využít s definicí CSS vlastnosti @font-face, jak ukazuje kód na další straně.

```
<head>
<link rel="preload" href="/fonts/awesome-l.woff2" as="
   font">
</head>

@font-face {
  font-family: 'Awesome Font';
  font-style: normal;
  font-weight: 400;
  font-display: auto; /* or block, swap, fallback,
     optional */
  src: local('Awesome Font'),
       url('/fonts/awesome-l.woff2') format('woff2'), /*
          will be~preloaded */
       url('/fonts/awesome-l.woff') format('woff'),
       url('/fonts/awesome-l.ttf') format('truetype'),
       url('/fonts/awesome-l.eot') format('
          embedded-opentype');
  unicode-range: U+000-5FF; /* Latin glyphs */
}
```

Optimalizace fontů v JavaScriptu

Pro zrychlení zobrazení textu můžeme využít některou z dostupných javascriptových knihoven. V principu nám dávají možnost reagovat na stav načtení fontu. Nejběžnějším způsobem je přidání class k elementu <html> po načtení fontu. Dojde tak k prohození písma. Mezi takové knihovny patří:

- Font Loading API,
- FontFace Observer,
- Font Load Events.

3.2.4 Minifikace HTML, CSS, JavaScript

Výraz „minifikace“ označuje proces odstraňování zbytečných znaků ve zdrojovém kódu. Mezi tyto znaky patří mezery, konce řádků, komentáře a oddělovače bloků, které jsou pro vývojáře užitečné, ale pro stroje jsou zbytečné. Minifikujeme zdrojové soubory HTML, CSS a JavaScript tak, aby je webový prohlížeč mohl číst rychleji [71].

3.3 Čas

3.3.1 LazyLoading

Jak napsal Martin Michálek, lazyloading označuje techniku, která zajistí načtení části obsahu stránky až ve chvíli, kdy ji uživatel potřebuje nebo bude potřebovat [22]. Nejčastěji je vázaný na posun stránky. Pokud se uživatel například dostane při posouvání stránkou na určitou hranici nad prvkem, začnou se teprve stahovat zdroje k danému prvku. Hojně se tato technika využívá k odkládání načítání obrázků, ale je vhodné ji využít i na další prvky, jako jsou například iframy. Jako dvě hlavní výhody lazyloadingu Martin Michálek uvádí:

- Uživatel ušetří data a vývojáři servery. Velká část lidí totiž neposune stránku a rovnou odchází nebo najde, co potřebuje, už v horní části stránky.
- Jako vývojáři můžeme pomocí lazyloadingu prioritizovat načtení jiných prvků.

Jak už bylo zmíněno výše, lazyloading je využíván především pro obrázky. Většina javascriptových knihoven využívá techniky, kdy není vložena cesta k souborům do parametru src nebo srcset, ale využijí tzv. data atributy, viz ukázka níže.

```
1 <img alt="small" data-src="obrazek.jpg">
```

Element <img> má jako jediný povinný atribut alt a element je takto validní. Prohlížeč si myslí, že není dána cesta k obrázku a tak nic nestahuje. Je potřeba ale

myslet na to, aby měl element definovanou minimálně výšku, jinak by docházelo k odskakování obsahu po načtení. JavaScript pak má následující úlohu:

- Čeká na událost, kdy se element dostane do viewportu.
- Vymění data atribut data-src za parametr src.

Prohlížeč následně obrázek stáhne a zobrazí tak, jak je běžné.

3.3.2 Asynchronous vs Deferred JavaScript

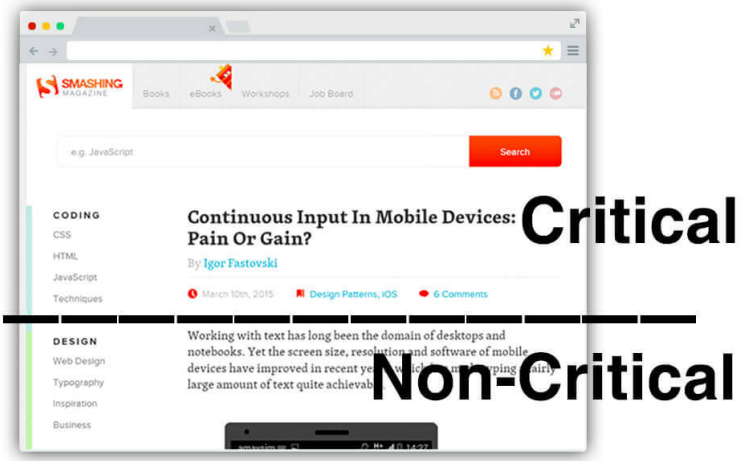
Pokud prohlížeč při parsování HTML narazí na element <script src="... /> zastaví parsování a stahuje daný script. To vede k pozdějšímu začátku vykreslování stránky a načtení. Způsob načítání JavaScriptu můžeme změnit dvěma způsoby jak uvádí Josef Žáček ve svém článku [61]:

- Atribut async - s atributem async je kód HTML parsován, zatímco prohlížeč načítá a zároveň spouští skript. K provedení skriptu může dojít, kdykoli je skript připraven (potenciálně před dokončením parsování HTML) poté, co byl načten paralelně s parsováním dokumentu.
- Atribut defer - s atributem defer je script paralelně stahován, zatímco se HTML parsuje. Nicméně pokud je script stáhnutý ještě před dokončením parsování HTML, script není spuštěn. Čeká se, až se celé HTML zpracuje.

3.3.3 Kritické CSS

Požadavek na CSS soubor může výrazně zvyšovat čas potřebný pro zobrazení obsahu webové stránky, jak uvádí Dean Hume [23]. Prohlížeč blokuje vykreslení stránky, dokud nenačte a nezpracuje všechny CSS soubory linkované v hlavičce stránky. Pokud tedy máme všechny styly v jednom CSS souboru, stránka se nezačne načítat, dokud se tento velký soubor celý nezpracuje. Jedná se o metodu nalezení minimálního CSS (kritického) potřebného k zobrazení viditelné části stránky, jak ukazuje obrázek 3.11. Toto kritické CSS se pak inline stylem zápisu přidá

do hlavičky stránky. Zbylé CSS, které je již umístěno v souboru, se pak pomocí JavaScriptu asynchroně dále načítá, zatímco uživateli se již vykresluje stránka.



Obrázek 3.9: Vymezení kritického CSS podle okna [23]

3.3.4 Kritický Javascript

Kritický JavaScript neboli JavaScript blokující načtení. Ještě předtím, než prohlížeč může vykreslit stránku, musí sestavit DOM a přeparsovat HTML. Během tohoto procesu, kdykoliv parser narazí na script, se parser zastaví a zpracovává ho. V případě externího scriptu ho musí stáhnout a dochází k dalšímu zdržení při načítání webové stránky. Cílem optimalizace je načítat JavaScript, který je nutný pro první vykreslení stránky, a až následně načítat asynchroně další scripty a externí zdroje [46].

3.3.5 Přednačítání - prefetching

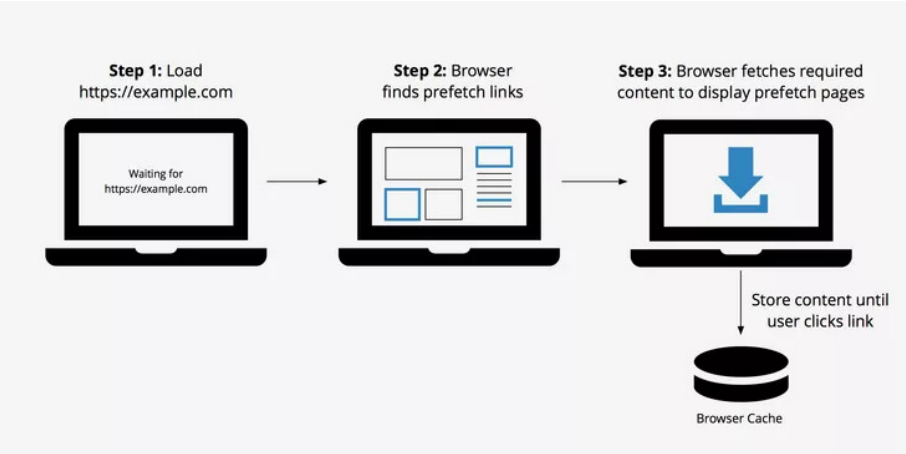
Prefetching spočívá v tom, že prohlížeč si stáhne potřebné zdroje (CSS, obrázky, JavaScript) ještě předtím, než jsou potřeba. Robin Rendle ve svém článku uvádí tyto typy prefetchingu [51]:

- DNS prefetching - informuje prohlížeč o URL, na kterou se bude dotazovat a umožňuje mu tak vyřešit DNS co nejrychleji.

- Preconnect - stejně jako předchozí vyřeší DNS, ale také provede TCP handshake a TLS.
- Prefetching - pokud jsme si jisti, že v budoucnu bude vyžadován určitý zdroj, můžeme požádat prohlížeč, aby si tuto položku vyžádal a uložil ji do mezipaměti pro pozdější použití.
- Prerendering pages - nám dává možnost předběžně načíst všechny zdroje z určité stránky. Je to jako otevření adresy URL na skryté kartě - stáhnou se všechny zdroje, vytvoří se DOM, stránka se načte, použije se CSS, spustí se JavaScript atd. [68].
- Preloading - na rozdíl od prefetchingu, které lze ignorovat, u preloadingu prohlížeč musí tento zdroj stáhnout a uložit pro pozdější použití.

Výsledkem je pak minimální doba pro načtení následující stránky. Jako jasný příklad se nabízí průchod košíkem eshopu. Jednotlivé kroky košíku jsou jasně definované. Mezitím, co uživatel vyplňuje například adresu pro zaslání zboží, už se mu načítají zdroje pro stránku s kompletním shrnutí objednávky. Ukázka kódu pro přednačtení obrázku:

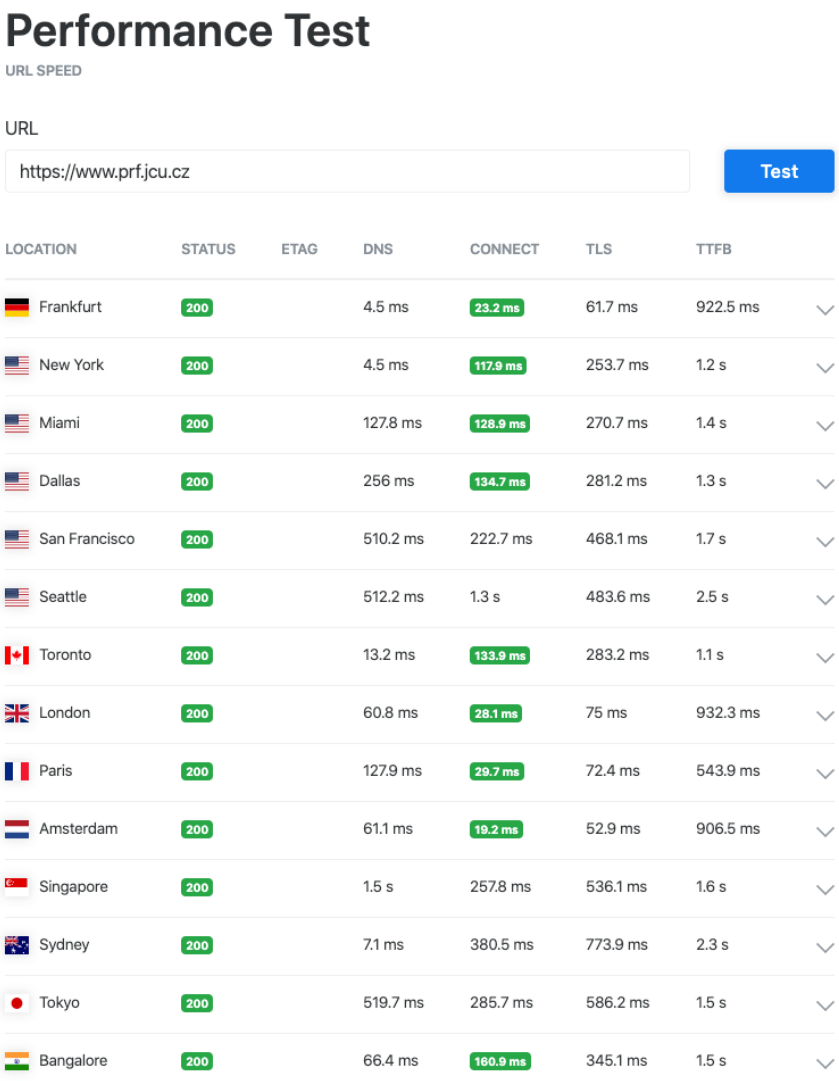
```
1 <link rel="prefetch" href="image.png">
```



Obrázek 3.10: Princip prefetchingu[52]

3.4 Server a zdroje

Na rychlost načtení webu má také veliký vliv místo, odkud na webovou stránku přicházíme a využití již předpřipravených souborů. Vliv polohy můžeme ověřit například pomocí nástroje Performance Test na webu keycdn.com, jak můžeme vidět na obrázku níže.



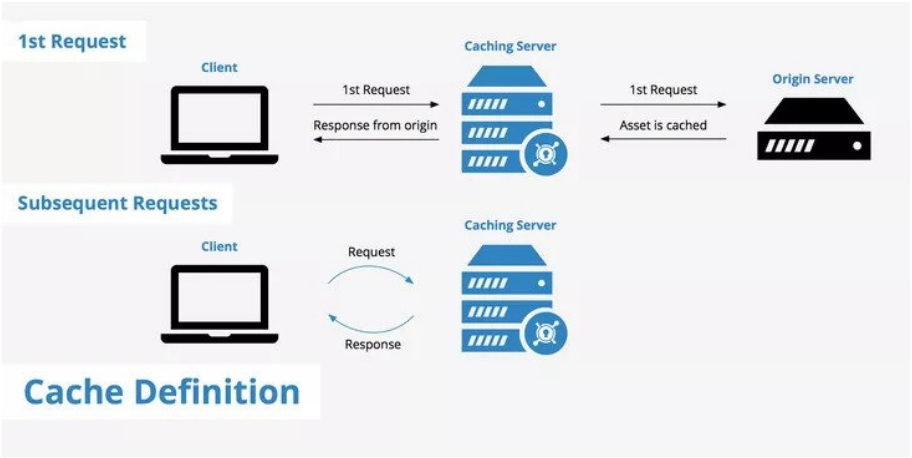
Obrázek 3.11: Vliv polohy na načtení webové prezentace *https://www.prf.jcu.cz/*

3.4.1 Cache

Obecně ukládání do mezipaměti (cache) pomáhá zlepšit rychlost přistupování ke zdroji, protože vytváří kopii zdroje a později přistupuje právě k této kopii uložené v mezipaměti namísto originálu [55]. Při optimalizaci frontendu nás zajímají především tyto dva typy cache.

Cache Server

Cache server je dedikovaný server nebo také edge server (v podmínkách CDN), který se používá v sítích pro poskytování obsahu nebo webových proxy serverů. Tyto servery mohou být umístěny v mnoha geografických oblastech a používají se k ukládání a poskytování dat tak, aby žádost / odpověď z prohlížeče nemusela putovat příliš daleko [55].



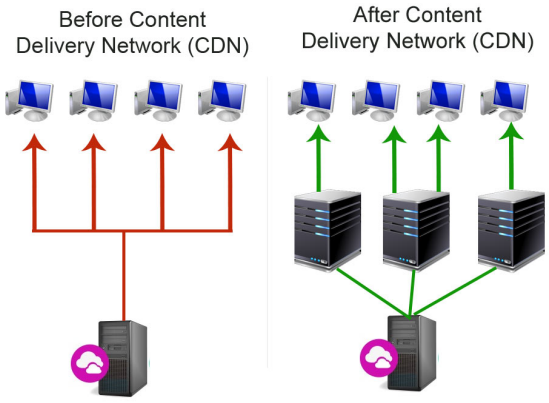
Obrázek 3.12: Princip cache serveru [55]

Browser Cache

Webové prohlížeče ukládají soubory ve své místní vyrovnávací paměti. Jsou tak přístupné rychleji, protože nemusí být stahovány ze serveru. Každý zdroj může definovat vlastní politiku ukládání do mezipaměti prostřednictvím záhlaví HTTP Cache-Control. Cache-Control záhlaví řídí, kdo ukládá odpověď, za jakých podmínek a na jak dlouho [56].

3.4.2 CDN

Síť pro doručování obsahu Content Delivery Network neboli CDN je síť počítačů vzájemně propojených skrze Internet, která umožňuje dostupnost obsahu nebo dat (obvykle velký multimediální obsah) uživatelům [44]. Jedná se o metodu, kdy jsou jednotlivé zdroje rozmístěny na různých místech a tím mají blíže ke koncové stanici, jak ukazuje obrázek níže.

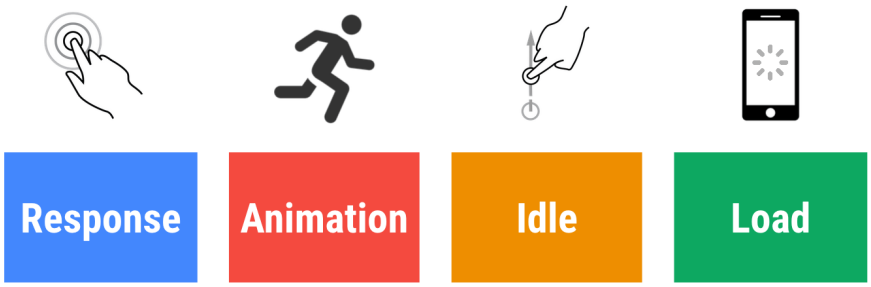


Obrázek 3.13: Způsob distribuce dat s a bez CDN sítě [45]

3.5 Měření rychlosti webové aplikace

3.5.1 Metodika RAIL

Model RAIL navrhl v roce 2015 Paul Irish [13]. Model je zaměřený na uživatele a rozděluje uživatelský zážitek na čtyři klíčové akce [43].



Obrázek 3.14: Čtyři klíčové akce podle RAIL model [43]

- Response - doba, za kterou webová aplikace reaguje na uživatelský vstup. Reakční doba aplikace by se měla držet pod 100 ms, aby interakce byla pro uživatele plynulá. Nejedná se o čas potřebný pro načtení aplikace.
- Animation - plynulost animací a průchod aplikací, tedy plynulost scrollování nebo přechody mezi prvky webové aplikace.
- Idle - popisuje dění na pozadí webové aplikace po načtení. Má za cíl maximalizovat dobu nečinnosti tak, aby stránka byla schopna reagovat na uživatelský vstup v rámci 50 ms.
- Load - je v kontextu RAIL modelu First Meaningful Paint neboli vyjadřuje, kdy bylo vykresleno na stránce něco smysluplného. Cílem RAIL modelu je, aby tento čas byl do jedné sekundy od požadavku od uživatele.

RAIL pomáhá vývojářům a designérům zajistit pro každou z těchto akcí dobrý uživatelský dojem za pomoci cílů a doporučení.

Cíle mohou sloužit jako metriky, které souvisejí s uživatelským zážitkem. Protože lidské vnímání je relativně konstantní, je nepravděpodobné, že by se tyto cíle nečekaně změnily.

Naopak doporučení, která vám pomohou dosáhnout cílů, mohou být specifická pro aktuální podmínky hardwaru a síťového připojení, a proto se mohou a budou časem měnit.

Cíle RAIL modelu tak jak je definovaly Meggin Kearney, Addy Osmani, Kayce Basques a Jason Miller [43]:

- Dokončit požadavek inicializovaný uživatelem do 100 ms.
- Vykreslovat při animacích každých 10 ms minimálně 1 snímek.
- Vizuální plynulost. Uživatelé si všimnou, když se obnovovací frekvence zpomalí.
- Maximalizovat využití nečinnosti aplikace ke zvýšení možnosti reakce na uživatelský vstup až do 50 ms.
- Optimalizace prostředků pro rychlejší načítání vzhledem k zařízení, na kterém je aplikace použita a síťových prostředků dostupných pro komunikaci. Jako cíl je definováno načtení celé stránky do 5 sekund nebo méně na průměrném mobilním zařízení připojené přes 3G síť.
- Pro následný přechod na další stránku je vytyčeným cílem načíst stránku za méně než 2 sekundy. Tento cíl se však může časem změnit.

Vnímání rychlosti podle RAIL

Rychlost webové aplikace je odlišná v závislosti na aktuálních podmínkách sítě a hardwaru. Tabulka níže zobrazuje, jakým způsobem uživatel vnímá rychlost načítání webové aplikace podle RAIL.

čas	vnímání uživatele
0 až 100 ms	aplikace je rychlá
100 až 300 ms	uživatelé zaznamenají mírné ale již zaznamenatelné zpoždění
300 až 1 000 ms	pro většinu uživatelů již tento čas představuje nahrávání nové stránky nebo její změnu
1 000 ms a více	hranice, kdy uživatelé ztrácí pozornost nad reakcí kterou očekávali
10 000 ms a více	hranice, kdy se uživatel cítí přímo frustrován, již neočekává reakci a s velkou pravděpodobností odchází pryč

Tabulka 3.1: Způsob vnímání rychlosti načítání webové aplikace podle RAIL

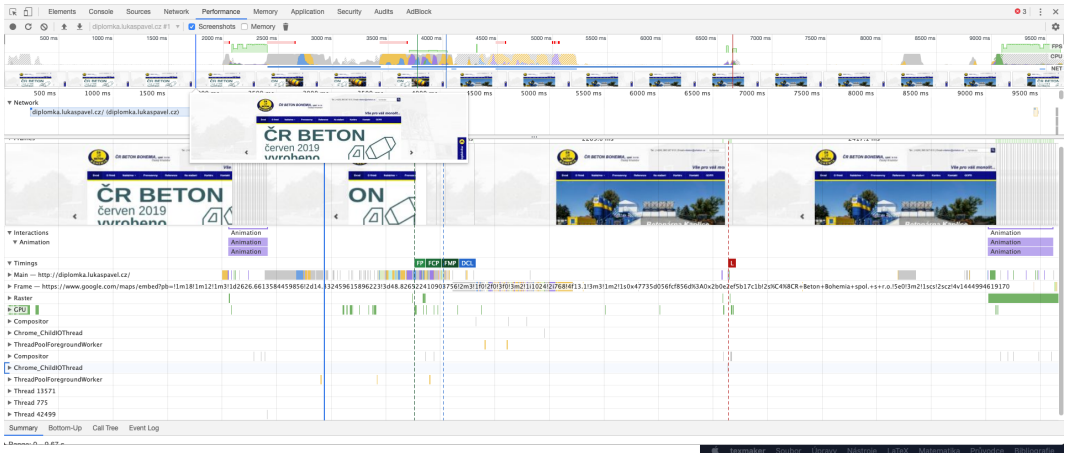
3.5.2 Nástroje pro měření rychlosti

Chrome Dev Tools

Chrome Dev Tools představuje panel, který je integrován do prohlížeče Google Chrome. Nabízí hloubkovou analýzu běhu a načítání webové aplikace.

V Chrome Dev Tools nás bude zajímat především panel performance. Kayce Basques ukazuje ve svém článku [39], jakým způsobem lze využít nástroje v tomto panelu. Jeho hlavní výhodou je schopnost měřit v reálném čase a poskytování okamžité zpětné vazby. Můžeme například sledovat FPS animací, monitorovat aktuální náročnost na CPU a další, jak ukazuje seznam níže:

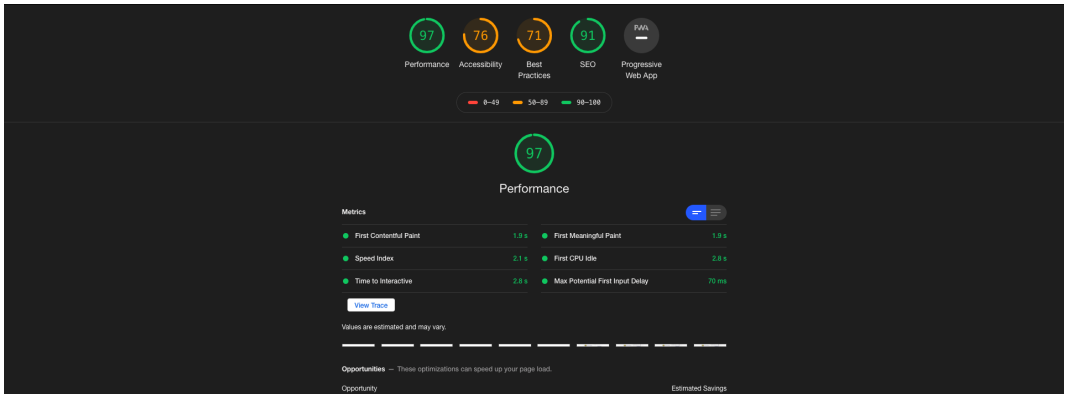
- Zpomalení CPU k nasimulování méně výkonného zařízení.
- Zpomalení sítě k nasimulování pomalejšího připojení k internetu.
- Zobrazení "main thread"neboli hlavního vlákna pro zachycení všech událostí, které nastanou během sledování.
- Zobrazení hlavního vlákna a jeho aktivit v tabulce, kterou lze řadit podle časové náročnosti.
- Analyzovat počet snímků za sekundu pro měření animací.
- Monitoring využití CPU a zpracovávání JavaScriptu.
- Analizovat požadavky na server během sledování/měření.
- Zaznamenávání screenshotů a nahrávání obrazu pro zpětné přehrání záznamu načtení a pohybu na stránce.
- Analýzu reakcí webové aplikace na interakci od uživatele.
- Nalezení problému s posouváním stránky v reálném čase.



Obrázek 3.15: Ukázka panelu Performance v Chrome Dev Tools

Lighthouse

Jedná se o nástroj od firmy Google a je k dispozici v Chrome Dev Tools, jako rozšíření prohlížeče Chrome, jako modul Node.js a lze ho integrovat do WebPageTest. Přiřadíte mu pouze URL adresu a on provede analýzu. Simuluje různá zařízení s variabilním připojením. Na stránce provede řadu auditů a poté vám poskytne zprávu o rychlosti načtení. Jeho hlavní výhodou je, že vývojáři poskytne návrhy na zlepšení a přímo ukazuje na problematické části aplikace. Poskytuje také audity, které zlepšují přístupnost, usnadňují údržbu stránky a další.

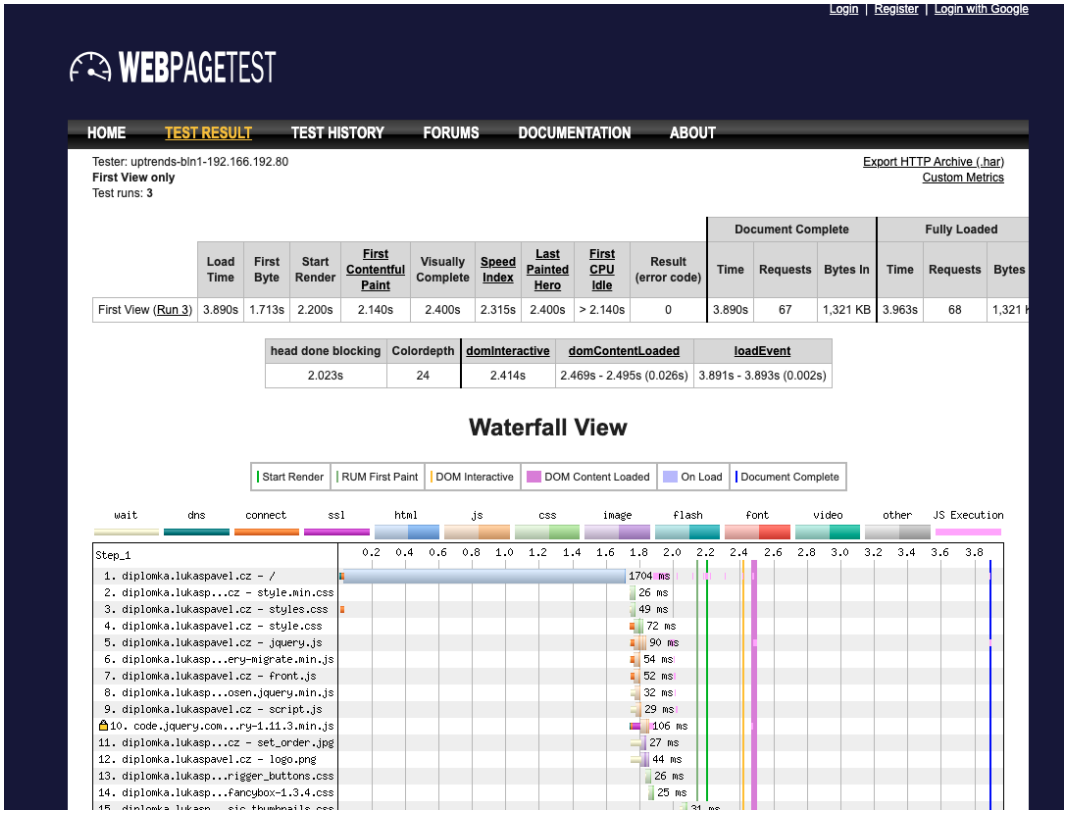


Obrázek 3.16: Ukázka výsledků auditu za pomoci Lighthouse

WebPageTest

WebPagetest je nástroj, který byl původně vyvinut společností AOL pro interní použití a byl uvolněn v roce 2008 na základě licence BSD. Nástroj se neustále vyvíjí a lze ho spustit na vlastním serveru. Online verze na www.webpagetest.org je provozována několika společnostmi a jednotlivci, kteří poskytují vlastní infrastrukturu po celém světě [58].

Mezi jeho hlavní přednosti patří možnost využití API a vlastních metrik. Dále do jeho nesporných výhod spadá velká komunita okolo nástroje a jeho dokumentace.

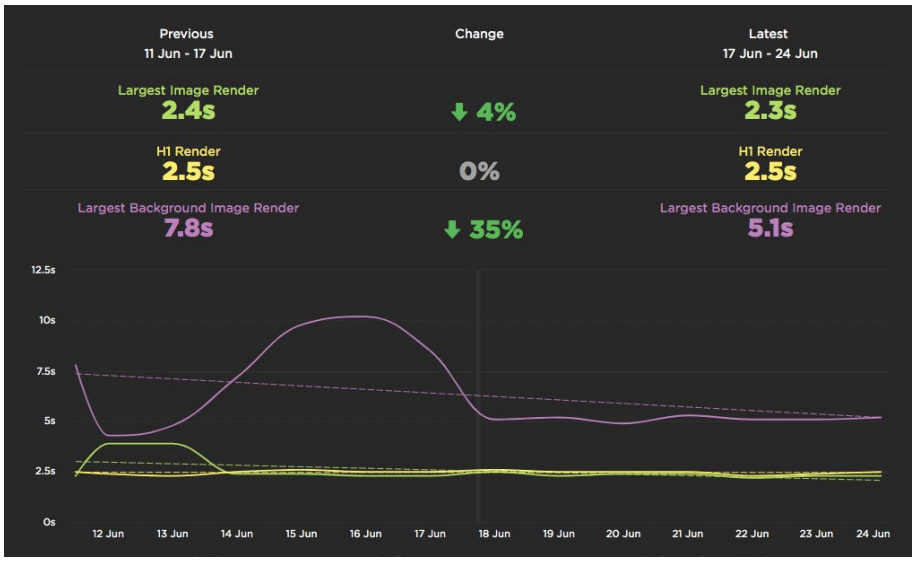


Obrázek 3.17: Ukázka online nástroje WebPageTest

Speedcurve.com

Jedná se o komerční nástroj, který nabízí podobné možnosti, jako již zmíněné nástroje. Jeho hlavní výhodou je možnost měření pomocí SpeedCurve LUX, který analyzuje reálné uživatele a tím sbírá RUM metriky. Další výhodou je automatické a pravidelné měření webu s možností upozorňování na změny a zobrazení

trendu v grafu. Rychlost načítání webu by měla být dlouhodobě sledována, pokud se obsah nebo zdrojový kód webu mění.



Obrázek 3.18: Ukázka grafu z nástroje speedcurve.com [59]

Porovnání nástrojů

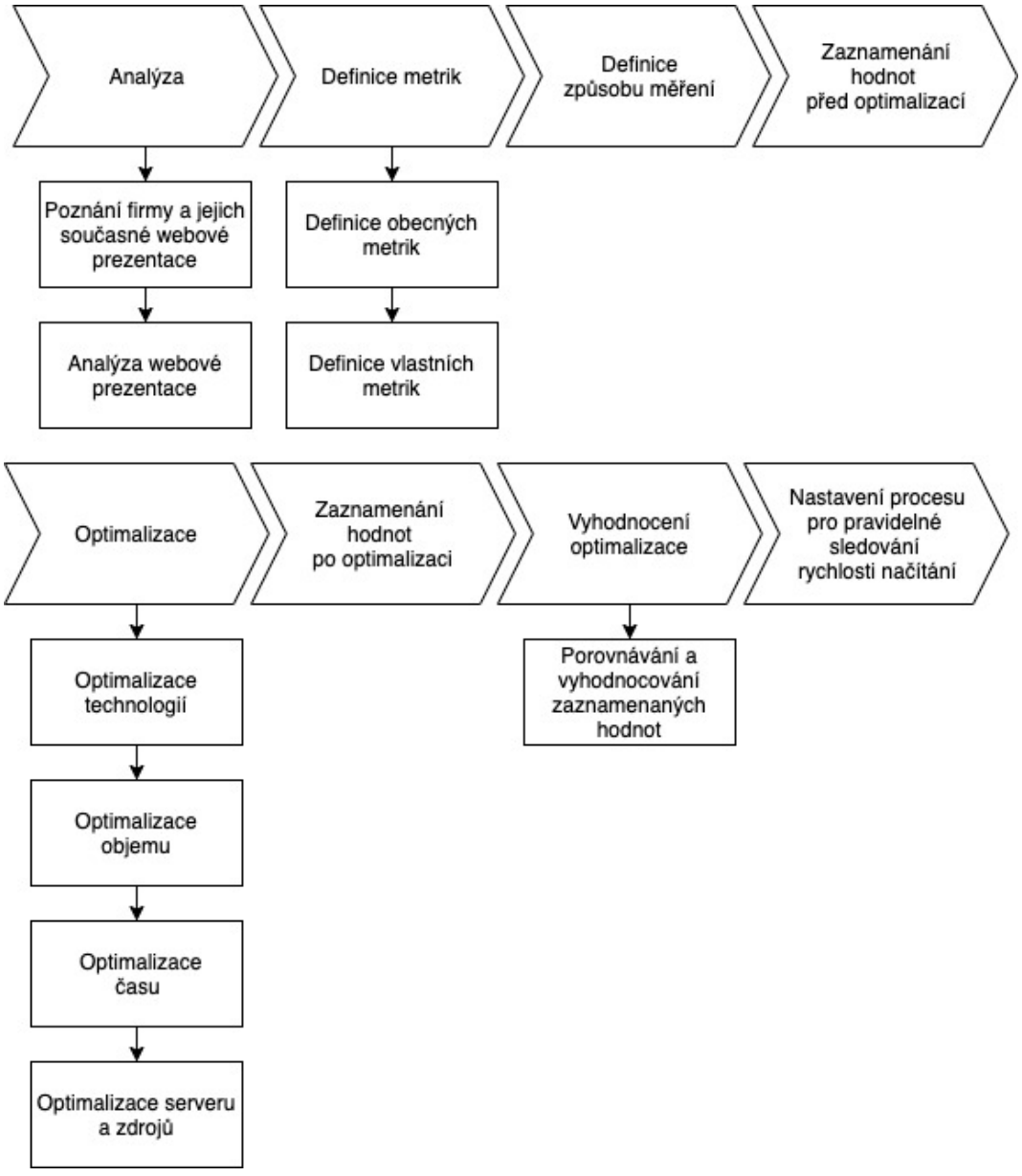
Každý z výše zmíněných nástrojů je vhodný pro určitý typ měření. Chrome Dev Tools je vhodný pro rychlé měření a ověření účinnosti provedených změn při optimalizaci. WebPageTest pro detailnější měření a archivaci výsledků pro další zkoumání. Lighthouse je vhodný jako nástroj pro vyhledávaní chyb a návrhů pro další optimalizaci. Speedcurve.com viniká v schopnosti monitorovat dlouhodobě rychlost načítání webové prezentace. Hlavní výhody a nevýhody shrnujeme v tabulce níže.

nástroj	hlavní výhoda	hlavní nevýhoda
Chrome Dev Tools	detailní měření v reálném čase a získávání RUM metrik	výsledky nelze ukládat pro pozdější analýzu
Lighthouse	konkrétní návrhy na zlepšení aplikace	menší možnosti nastavení
WebPageTest	webPageTest API, videa z načítání stránky, vlastní metriky	čekání na zpracování testu
Speedcurve.com	měření RUM metrik, vizualizace	placený nástroj

Tabulka 3.2: Porovnání výhod a nevýhod výše zmíněných nástrojů

4. Metodika optimalizace webové prezentace

Metodika je zaměřená na poznání firmy, její webové prezentace a následná optimalizace a měření na základě analýzy uživatelů. Schéma metodiky můžeme vidět na obrázku 4.1.



Obrázek 4.1: Schéma metodiky

Hlavním cíle metodiky je optimalizace rychlosti načítání vzhledem k zařízení, ze kterého uživatel přistupuje a jeho připojení k síti. Jako cíl je definováno načtení stránky (metrika Load Event) do 5 sekund nebo méně na průměrném mobilním zařízení připojené přes mobilní síť a do 1 sekundy na průměrném stolním počítači s domácím WiFi připojením.

4.1 Poznání firmy a jejich současné webové prezentace

Prvním krokem, ještě před samotnou optimalizací webové prezentace, by mělo být seznámení se s firmou a poznání jejich současné webové prezentace. Ta může být na odlišné úrovni a firma k ní může mít různé podklady. V zásadě bychom se měli zajímat o funkci a nastavené cíle webové prezentace pro firmu. Dále bychom se měli zajímat, zda již je nějakým způsobem monitorován, případně vyhodnocován provoz. V této fázi bychom měli vědět:

- Základní informace o firmě a oblasti jejího působení.
- Zda existují podklady, které byly použity při tvorbě webové prezentace (analýza konkurence, architektura webu, cílový zákazník, definice cílů, atd.).
- Jaké jsou cíle webové prezentace.
- Zda je webová prezentace měřena nebo vyhodnocována analytickým nástrojem.

4.1.1 Analýza webové prezentace

Druhým krokem by měla být analýza současné webové prezentace a její vyhodnocení. Jedná se o jeden z nejdůležitějších kroků v rámci celé optimalizace, protože dochází ke zjištění, jakým způsobem je web používán, kdo ho používá a jaký je nejčastější způsob průchodu webem. Pro potřeby optimalizace rychlosti načítání potřebujeme znát odpovědi na následující otázky:

Z jakého zařízení uživatelé přistupují na webovou prezentaci?

Zajímáme se o podíl mobilního zařízení a stolního počítače, o velikost rozlišení a využívaný operační systém. Na základě odpovědí na tuto otázku následně budeme definovat zaměření optimalizace.

Z jaké polohy se uživatelé připojují?

Zde se potřebujeme analyzovat geografickou oblast, ze které uživatelé přistupují na webovou prezentaci. Následně podle toho určíme, z jakého bodu budeme měřit výsledky optimalizace nebo zda musíme využít několika bodů a opakovat měření.

Jaké prohlížeče a jejich verze využívají pro připojení?

Každý web má svou cílovou skupinu a ta se může lišit v tom, jaké prohlížeče a verze používají pro zobrazení webové prezentace. Odpověď nám řekne, jaké metody můžeme využít při optimalizaci a zda musíme dbát na to, aby provedené úpravy byly zpětně kompatibilní s některými prohlížeči a jejich verzemi.

Jaké stránky uživatelé navštěvují?

Zkoumáme, jaká je vstupní stránka webu a jaký je průchod webovou prezentací. Zde potřebujeme definovat, na jakou stránku se při optimalizaci zaměřit a případně jakým způsobem lze usnadnit další průchod webovou prezentací. Využijeme to například při definici zdrojů, které je vhodné přednačíst pro další stránku po načtení současné.

Jaké jsou hlavní cíle webové prezentace?

Jedná se o nejdůležitější otázku ze všech, jelikož nám často udává i cíl samotné optimalizace. Hlavní cíle jsou zpravidla i podkladem pro definici vlastních metrik. Velice často odpověď na tuto otázku získáme z podkladů, které byly použity při vytváření webové prezentace. Nicméně je zde důležité udělat opětovnou validaci takto definovaných cílů na základě zkušeností z provozu.

Jaké jsou technologie použité na webové prezentaci?

Následně analyzujeme současné technologie použité na webové prezentaci. Především v jakém jazyce je napsaná, zda se jedná o statický web nebo o web využívající CMS systém. Dále zda je umístěn na vlastním serveru, jeho nastavení nebo zda firma využívá webhostingových služeb externí firmy a jaké má parametry.

4.2 Definice obecných metrik

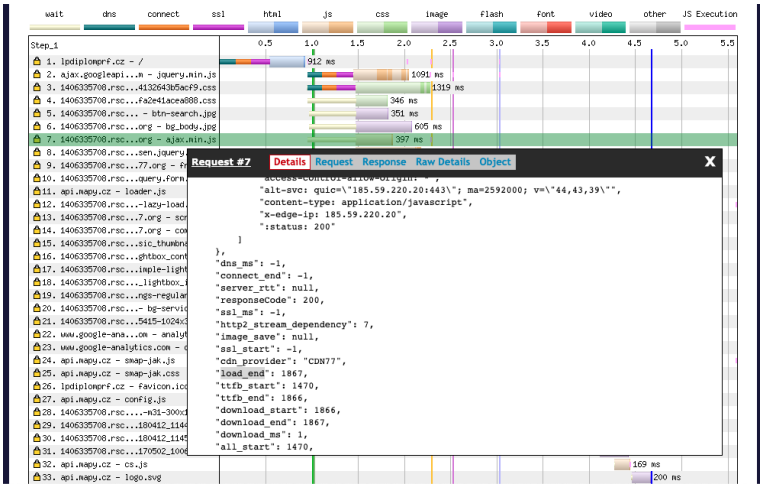
Na základě předešlé analýzy bychom měli sestavit metriky, které budeme v rámci optimalizace měřit a vyhodnocovat. Níže uvádíme seznam základních metrik:

- TTFB - metrika, na jejíž základě budeme sledovat rychlost serveru.
- DOM Loading - metrika, na jejíž základě budeme sledovat začátek zpracovávání HTML stránky.
- DOM Interactive - metrika, na jejíž základě budeme sledovat dokončení zpracovávání HTML stránky.
- DOM Complete - metrika, na jejíž základě budeme sledovat dokončení zpracovávání HTML stránky a všech blokujících zdrojů.
- Load Event - metrika, na jejíž základě budeme sledovat kompletní dokončení nahrávání stránky.
- Speed Index - obecná metrika, na jejíž základě budeme sledovat celkový dopad optimalizace.
- Start render - obecná metrika, podle které budeme zjišťovat první vykreslení prvku na stránce.
- First Meaningful Paint - obecná metrika, podle které budeme zjišťovat první smysluplné vykreslení stránky.
- Time to First Interactive - obecná metrika, na jejíž základě budeme zjišťovat, kdy je poprvé stránka připravena na vstup od uživatele.

- Velikost CSS - obecná metrika pro velikost stylů jakožto jednoho ze zdrojů, který musí prohlížeč zpracovat a může tak blokovat načtení celé stránky.
- Velikost fontů - obecná metrika pro velikost zdrojových souborů pro písmo.
- Velikost JavaScriptu - obecná metrika pro velikost JavaScriptu jako dalšího zdroje.
- Velikost obrázků - obecná metrika pro velikost přenášených obrázků, jelikož se často jedná o největší zátěž pro přenos z pohledu objemu dat.

4.3 Definice vlastních metrik

Vlastní metriky jsou při optimalizaci velmi často opomíjeny a přitom se jedná o nejdůležitější metriky z pohledu firmy respektive zadavatele. Vlastní metriky jsou definovány na základě poznání firmy, analýzy webové prezentace a především hlavních cílů prezentace. Jedná se o rozšíření standardních událostí v prohlížeči. Jde o metriky, které jsou definovány jako načtení určitého prvku na obrazovce uživatele nebo jako čas, který je potřebný pro stáhnutí určitého zdroje. Metriky, které jsou definované jako zobrazení určitého prvku. lze měřit pomocí Element Timing API. Metriky, které definují stažení určitých zdrojů můžeme vyhodnocovat pomocí analýzy requestu na daný zdroj a jeho atributu load end, jak ukazuje obrázek níže.



Obrázek 4.2: Analýza requestu v nástroji WebPageTest

Jako nejčastější definované vlastní metriky lze určit:

- Zobrazení loga - pro základní rozpoznání stránky a šíření značky.
- Zobrazení navigace - jako jednoho z nejdůležitějších prvků pro uživatele pro další postup skrz webovou prezentaci.
- Zobrazení nadpisu - pro orientaci uživatele na stránce.
- Zobrazení prvního smysluplného textu - pro určitý typ stránek jedna z hlavních metrik, kdy uživatel může začít číst text, například blogy nebo firemní webovou prezentaci.
- Zobrazení kontaktu - jako jeden z nejčastějších cílů firemních webových prezentací.
- Zobrazení vyhledávání - důležitý prvek pro uživatele zejména na eshopech a katalozích.
- Zobrazení interaktivního prvku - jako jednoho z častých cílů webové prezentace, neboť jejich prostřednictvím získáváme informace od uživatele.
- Zobrazení obrázku produktu - jeden z důležitých prvků pro eshopy na detailu produktu.
- Zobrazení tlačítka pro přidání do košíku - nejdůležitější prvek pro eshopy.
- Zobrazení tlačítka pro dokončení nebo pokračování v košíku - důležitý prvek pro uskutečnění konverze pro eshopy.
- Stáhnutí CSS stylů - pro webové prezentace, které obsahují hodně animací nebo jsou rozmanité, se může jednat o kritický zdroj.
- Stáhnutí JavaScriptu - stejně jako u CSS stylů se může jednat o kritický zdroj pro zahájení animace a zvýšení uživatelského prožitku.

4.4 Definice způsobu měření

Další část definované metodiky spočívá ve způsobu a vlastnostech měření. Měření musí být provedeno zvlášť pro mobilní zařízení a zvlášť pro stolní počítač s odpovídající rychlostí připojení, latencí a pokud je to možné, také s odpovídajícím rozlišením na základě přístupů z analýzy webové prezentace. Zároveň by měření mělo probíhat na několika stránkách webu podle jejich typu tak, aby pokrylo většinu navštěvovaných stránek.

Pro rychlé ověřování efektu prováděných změn lze využívat nástroj Chrome Dev Tools a jeho záložku performance. Pro návrhy, jaké způsoby optimalizace využít a kde jsou slabiny v rychlosti načítání, využijeme nástroj Lighthouse. Pro měření metrik a hledání dalších problémů při načítání využijeme nástroj WebPageTest.

Vyhodnocování metrik budeme získávat na základě několikanásobného měření a určení jejich střední hodnoty ze všech uskutečněných měření. Velice často se stává, že vývojář vyhodnocuje optimalizaci pouze na základě jednotlivých měření a je tak klamán, neboť jednotlivá měření se mohou výrazně lišit v závislosti na kvalitě spojení nebo zatížení serveru.

Důležitou vlastností měření je, že vychází z analýzy webové prezentace a je upraveno pro konkrétní užití. V použité metodice nastavujeme rychlost připojení, latenci, rozlišení, typ zařízení a případné zpomalení CPU na všech využívaných nástrojích, pokud je tato nastavení umožňují.

4.5 Zaznamenání hodnot před optimalizací

Před samotnou optimalizací si změříme podle definovaného způsobu měření výchozí hodnoty, abychom následně mohli porovnávat její úspěšnost. Naměřené hodnoty zaznamenáme do tabulky v nástroji Google Sheets.

4.6 Optimalizace

Následuje proces optimalizace webu, jehož součástí je opakované měření pomocí nástroje Google Dev Tools pro ověření účinnosti optimalizačních metod.

4.6.1 Optimalizace technologií

V první části optimalizace se zaměříme na použité technologie a jejich aktualizaci. Začneme s tímto krokem, jelikož má globální efekt na celou webovou prezentaci.

Přechod z HTTP na HTTP/2

Jak bylo již zmíněno v kapitole 3.1 HTTP/2 se soustřeďuje na výkonnost, snížení latence u přenosu a snižování zátěže na serveru a síti. Servery, ale i webhostingové služby již v drtivé většině tuto možnost nabízejí. Zda webová prezentace využívá k přenosu zdroje protokol HTTP/2, zjistíme v nástroji Chrome Dev Tools v záložce *Network* ve sloupci *Protocol*. Ten v případě přenosu přes HTTP2 obsahuje hodnotu *h2*.

Aktualizace CMS a využitých rozšíření

Pokud webová prezentace využívá některý z CMS systémů, provedeme jeho aktualizaci společně s dalšími pluginy nebo rozšířeními, které používá.

Aktualizace externích knihoven

V posledním kroku této části analyzujeme ostatní externí knihovny, které využívá webová prezentace. Zde ale musíme být velice opatrní, jelikož jejich aktualizace může naopak zvýšit náročnost z pohledu objemu přenášených dat. Porovnáváme jednotlivé aspekty aktualizace a jejího přínosu z pohledu bezpečnosti, výkonnosti (zatížení procesoru), objemu dat pro přenos. Zároveň pokud jsou nahrávány z externího zdroje, zkontrolujeme, že jsou přenášeny přes HTTP/2 protokol stejným způsobem, jako při přechodu z HTTP na HTTP/2.

HTTP komprese

Jedná se o komprimaci HTTP requestů a přenášených zdrojů. Zda jsou přenášené soubory komprimovány ,zjistíme v nástroji Chrome Dev Tools v záložce *Network* ve sloupci *Content-Encoding*. U každého souboru vidíme název nebo zkratku použité komprimace. Pokud je pole prázdné, soubor není komprimován. Ve většině případů kompresi povolíme přidáním kódu do souboru *.htaccess*.

4.6.2 Optimalizace objemu

V druhé části optimalizace se zaměříme na snižování množství dat potřebných pro kompletní načtení webové stránky přenesené přes síť.

Nevyužité zdroje a duplikace

Pokračujeme analýzou nahrávaných JavaScript souborů a CSS stylů. Velice často se stává, že na stránce načítáme nepotřebné zdroje nebo je dokonce duplikujeme. V této části tedy provedeme analýzu requestů v nástroji *Chrome Dev Tools*. Tím zjistíme jaké scripty, knihovny a styly jsou stahovány pro danou stránku. Následně nepotřebné zdroje a duplikace odstraníme.

Optimalizace obrázků

Následuje optimalizace obrázků, jejich rozměrů, datové velikosti a formátu. K analýze problematických obrázků nám pomůže nástroj WebPageTest a záložka *Performance review* na výsledku testu. Na této záložce vidíme sekci *Compress Images* a seznam problematických obrázků. Pro jednotlivé návrhy pro optimalizaci nám poté poslouží záložka *Image Analysis*, která nám nabídne detailní porovnání jednotlivých formátů a jejich možné stažení.

Pro velký počet obrázků, které nahrává uživatel, například fotogalerie, je výhodnější optimalizaci více automatizovat za pomoci stáhnutí zdrojových souborů, jejich optimalizace a poté nahrazení přímo na serveru. Vhodný nástroj pro takovou optimalizaci je *ImageOptim* nebo *FileOptimizer*. Následně, pokud je to možné, zavedeme automatickou optimalizaci při nahrávání na server.

Optimalizace fontů

V další části se zaměříme na webové fonty. Zajímáme se, zda jsou opravdu využívané na stránce, zda jsou dostupné všechny formáty a jakým způsobem jsou načítány, viz kapitola 3.2.3.

Minifikace HTML, CSS, JavaScript

Na konci optimalizace objemu provedeme minifikaci HTML, CSS a JavaScript souborů. K tomu můžeme využít jeden z mnoha online nástrojů, využít některý z buildovacích nástrojů viz kapitola 1.4. nebo některého z CMS rozšíření.

4.6.3 Optimalizace času

V třetí části optimalizace se zaměříme na zrychlování vykreslení stránky a odkládání načtení dalších zdrojů. Pro analýzu způsobu vykreslení stránky v čase nám poslouží nástroj WebPagetest a jeho *Filmstrip View*, na kterém můžeme vidět postupné vykreslení stránky v čase společně s osou aktuálně nahrávaných zdrojů.

LazyLoading

Začneme odkládáním načítání obrázků mimo obrazovku. V prohlížeči Chrome ve verzi 76 je možné nastavit lazyloading přímo v HTML pomocí atributu *loading*. Případně využijeme jednu z mnoha JavaSriptových knihoven, jako například *MiniLazyload* nebo některý z pluginů pro CMS.

Async, Defer a kritické zdroje

Poté upravíme způsob načítání JavaScriptu a CSS stylů tak, aby neblokovaly vykreslování pomocí async a defer atributů. Následně vytvoříme kritický JavaScript a CSS styly, které vložíme do kritické cesty tak, aby se načetly ještě před samotným začátkem vykreslování.

Prefetching

Zdroje, které jsme přesunuli z kritické cesty, přednačteme pomocí prefetchingu. Stejně tak na základě analýzy webové prezentace přednačteme scripty a styly pro další stránku.

4.6.4 Optimalizace serveru a zdrojů

Ve čtvrté části optimalizace se zaměříme na optimalizaci serveru a jeho rychlosti odpovědi. Spočívá v cachování souborů a přiblížením externích zdrojů co nejblíže uživateli.

Cache

Využijeme nastavení cache tak, aby server nemusel znovu zpracovávat stejné požadavky. K tomu využijeme browser cache pro obrázky, CSS styly, JavaScript a ico soubory. Optimalizace spočívá v úpravě *.htaccess* souboru.

Pro analýzu nastavení HTTP hlaviček využijeme Chrome Dev Tools a jeho záložku *Network*. Po zobrazení detailu zdroje vidíme nastavení parametru cache-control v *Response Headers*.

CDN server

V rámci optimalizace zdrojů umístíme všechny externí zdroje (obrázky, CSS styly, JavaScript a ico soubory) na CDN server. Tímto krokem snížíme zátěž na severu a zároveň i čas potřebný k zahájení stahování zdroje. Ve většině případů se jedná o placenou službu, i když některé CDN nabízejí omezené balíčky zdarma.

Pro analýzu využijeme nástroj WebPagetest a záložku *Performance Review*, na které můžeme vidět jednotlivé zdroje a sloupec *CDN detected*.

4.7 Zaznamenání hodnot po optimalizaci

Po ukončení optimalizační části provedeme měření výstupních hodnot optimalizace podle definovaného způsobu měření stejně jako tomu bylo v části zaznamenání hodnot před optimalizací.

4.8 Vyhodnocení optimalizace

Následně naměřené hodnoty před a po optimalizaci porovnáme a vypočítáme, o kolik procent se zvýšila nebo snížila každá jednotlivá metrika na všech měřených stránkách. Poté uděláme průměr pro jednotlivé metriky, abychom zjistili, o kolik procent se v průměru změnila každá metrika. Zároveň uděláme průměr všech metrik, abychom zjistili celkovou průměrnou změnu rychlosti načítání.

Následně z naměřených hodnot po optimalizaci zkontrolujeme, zda optimalizace splnila hlavní cíl metodiky.

4.9 Nastavení procesu pro pravidelné sledování rychlosti načítání

Pokud se jedná o dynamickou webovou prezentaci, po zhodnocení optimalizace nastavíme proces, který bude pravidelně sledovat rychlost webu a umožní nám reagovat na změny.

5. Optimalizace webové prezentace pro firmu ČR Beton Bohemia spol. s r.o.

5.1 Poznání firmy a jejich současné webové prezentace

V praktické části diplomové práce jsme optimalizovali webovou prezentaci společnosti ČR Beton Bohemia spol. s r.o., která je součástí nadnárodní skupiny Kirchdorfer Gruppe a za dobu své existence se stala významným obchodním partnerem stavebních firem v oblasti jižních a části severních Čech. Svým zákazníkům nabízejí komplexní služby v oblasti:

- transportbetonu,
- výroby a ukládky betonářské výztuže a armatury,
- monolitických konstrukcí,
- průmyslových podlah.

Obrázek 5.1: Hlavní strana webové prezentace firmy ČR Beton Bohemia spol. s r.o.

5.1.1 Analýza webové prezentace

Pro potřeby analýzy webové prezentace jsme použili data z nástroje Google Analytics za poslední tři měsíce. V rámci zachování firemního tajemství nebylo povoleno publikovat konkrétní čísla, ale pouze procentuální rozložení. Vzorek pro analýzu byl v rámci jednotek tisíc přístupů.

Z jakého zařízení uživatelé přistupují na webovou prezentaci?

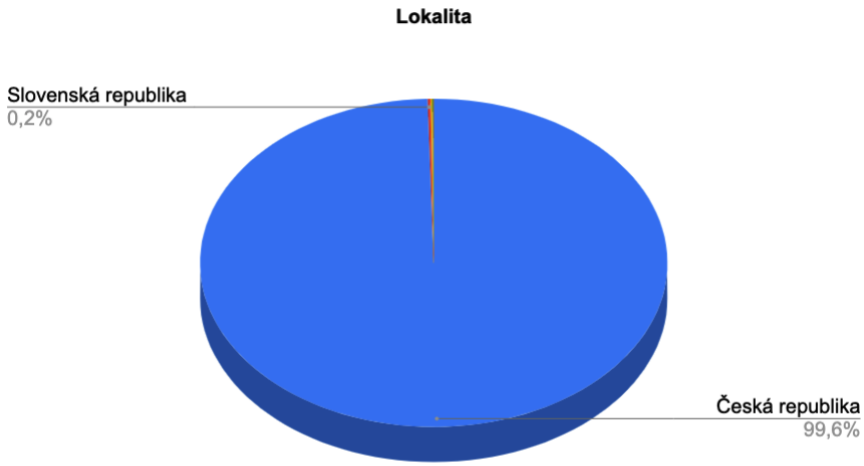
Využili jsme metriky *návštěvy podle zařízení*, viz obrázek 5.2. Dále jsme analyzovali velikost rozlišení, které mají uživatelé ve spojení s použitým operačním systémem, viz obrázek 5.3. Na základě těchto metrik můžeme definovat, že většina přístupů se uskutečňuje přes mobilní zařízení s rozlišením 360px x 640px a více. Pokud uživatelé přistupují ze stolního počítače, tak nejčastěji s rozlišením 1920px x 1080px.

Obrázek 5.2: Návštěvy podle zařízení z nástroje Google Analytics

Obrázek 5.3: Rozlišení zařízení spojené s operačním systémem uživatelů webové prezentace z nástroje Google Analytics

Z jaké polohy se uživatelé připojují?

K zodpovězení této otázky jsme využili analýzu *lokality*. Na základě této metriky jsme definovali, že uživatelé se připojují v zásadě pouze z České republiky, jak je vidět na obrázku 5.4.



Obrázek 5.4: Analýza lokality, ze které se uživatelé připojují

Jaké prohlížeče a jejich verze využívají pro připojení?

Zde jsme analyzovali metriku *prohlížeč a operační systém*. Následně jsme analyzovali i jednotlivé verze prohlížečů. Podle výsledků můžeme konstatovat, že ve více než polovině přístupů uživatelé přistupují z prohlížeče Google Chrome a využívají novější verze prohlížečů, jak ukazuje obrázek 5.5.

Obrázek 5.5: Prohlížeče, přes které uživatelé navštěvují webovou prezentaci

Jaké stránky uživatelé navštěvují?

U této otázky jsme analyzovali přístupy na jednotlivé stránky webové prezentace pomocí *analýzy chování*. Z výsledků je patrné, že hlavní strana je nejčastější vstupní stránkou celé webové prezentace. Zároveň je ale důležitá stránka "Ke stažení", následována textovými stránkami a provozovnami, viz obrázek 5.6 na další straně.

Obrázek 5.6: Počet přístupů podle typu stránky

Jaké jsou hlavní cíle webové prezentace?

Na setkání se zástupci firmy jsme diskutovali nad definovanými cíli, které byly definovány již při vytváření webové prezentace. Tyto cíle se nemění a zůstávají stejné. Jedná se o:

- Informování uživatele o službách poskytovaných společností ČR Beton Bohemia spol. s r.o.
- Zobrazení kontaktů.

Jaké jsou technologie použité na webové prezentaci?

Web je umístěn na pronajatém webhostingovém řešení. Server má nainstalované PHP ve verzi 7.3.11. Pro přenos dat ze serveru je využíván protokol HTTP ve verzi 1. Webová prezentace využívá CMS Wordpress ve verzi 4.8.11, který je napsán v jazyce PHP. CMS Wordpress má nainstalované následující pluginy:

- Accordions,
- Adminimize,
- Akismet Anti-Spam,

- All 404 Redirect to Homepage,
- All In One SEO,
- Contact Form 7,
- CPT Bootstrap Carousel,
- Download Manager,
- Filenames to latin,
- Intuitive Custom Post Order,
- Loco Translate,
- Nav Menu Roles,
- NextGEN Gallery,
- Simple Map,
- UpdraftPlus - Backup/Restore,
- Wordfence Security,
- WPDM - Extended Short-codes,
- WPDM - TinyMce Button.

Jako aktivní šablona je v CMS Wordpress zvolena šablona na míru, která využívá CSS framework *Bootstrap* a CSS styly má napsané v preprocesoru *LESS*. Vše je následně kompilováno pomocí buildovacího nástroje *Gulp*.

5.2 Definice obecných metrik

Z obecných metrik jsme sledovali na webové prezentaci základní definované metriky podle metodiky.

5.3 Definice vlastních metrik

Na základě cílů webové prezentace, analýzy chování uživatelů a návštěvnosti jednotlivých stránek jsme určili následující vlastní metriky, které jsme při optimalizaci sledovali:

- zobrazení loga,
- zobrazení kontaktu v hlavičce,
- zobrazení menu,
- zobrazení služeb pod sliderem na hlavní straně,
- zobrazení nadpisu,
- zobrazení tlačítka pro stažení souboru na stránce ke stažení.

5.4 Definice způsobu měření

Měřili jsme pomocí několika nástrojů a využili jejich hlavních výhod. Pro rychlé ověřování efektu prováděných změn jsme využívali nástroj *Chrome Dev Tools*. Na další návrhy, jak optimalizovat webovou prezentaci, nástroj *Lighthouse*. Pro komplexní měření jsme využívali online nástroj *WebPageTest*. Kompletní měření a vyhodnocení jsme dělali pomocí nástroje *Sitespeed.io*.

U nástroje Sitespeed.io jsme využili jeho napojení na *WebPageTest API*, které vygenerovaný report uloží na lokální disk. Díky tomu jsme automatizovali analýzu více stránek a opakovaného měření najednou. Tento report obsahuje i zdroj reportu v HAR formátu, který jsme následně zpracovávali a vyhodnocovali pomocí scriptu.

Měření proběhlo jak na emulátoru mobilního zařízení, tak stolního počítače. Z analýzy rozlišení uživatelů jsme definovali jako výchozí rozlišení pro testování na mobilním zařízení 360px x 640px a pro stolní počítač rozlišení 1920px x 1080px.

Podle webu dls.cz byla v lednu roku 2019 průměrná rychlost mobilní sítě 3G 8,14 Mb/s a u pevného připojení 20,47Mb/s [60], což při měření bylo simulováno.

Konkrétně u mobilního zařízení se jedná o připojení 9 Mb/s down and up, 170 ms latence a u stolního počítače připojení 20 Mb/s down, 5 Mb/s up, 20 ms latence.

Na základě analýzy typu stránek podle návštěvnosti a definicí metrik proběhlo měření na těchto stránkách:

- hlavní strana,
- stránka Transportbeton jako reprezentace textové stránky,
- stránka Ke stažení,
- stránka Provozovny,
- stránka provozovna Betonárna – Český Krumlov, jako reprezentace detailu provozovny.

5.5 Zaznamenání hodnot před optimalizací

Metrika	Hlavní strana	Transportbeton	Ke stažení	Provozovny	Provozovna Český Krumlov
TTFB	765 ms	1106 ms	863 ms	760 ms	723 ms
DOM Loading	791 ms	1131 ms	886 ms	783 ms	753 ms
DOM Interactive	3031 ms	2650 ms	2512 ms	2618 ms	2897 ms
DOM Complete	5599 ms	3469 ms	3238 ms	4742 ms	3260 ms
Load Event	5602 ms	3472 ms	3240 ms	4745 ms	3262 ms
Speed Index	2934 ms	2716 ms	2625 ms	2535 ms	2531 ms
Start render	2422 ms	2617 ms	2514 ms	2300 ms	2385 ms
First Meaningful Paint	2485 ms	2660 ms	2572 ms	2344 ms	2432 ms
Time to First Interactive	11450 ms	3372 ms	3568 ms	4607 ms	4638 ms
Velikost HTML	39.83 KB	23.12 KB	41.64 KB	35.50 KB	24.64 KB
Velikost CSS	147.52 KB	114.49 KB	114.49 KB	115.39 KB	114.49 KB
Velikost fontů	18.03 KB	0.00 KB	0.00 KB	33.25 KB	0.00 KB
Velikost obrázků	529.93 KB	95.51 KB	102.33 KB	744.45 KB	111.48 KB
Kontakt v hlavičce	2500 ms	2634 ms	2567 ms	2334 ms	2412 ms
Logo	2500 ms	2634 ms	2567 ms	2334 ms	2412 ms
Zobrazení menu	2500 ms	2634 ms	2567 ms	2334 ms	2423 ms
Zobrazení nadpisu	3420 ms	2645 ms	2567 ms	2334 ms	2423 ms
Zobrazení tlačítka pro stažení	x	x	2750 ms	x	x
Zobrazení služeb na HP	3012 ms	x	x	x	x

Tabulka 5.1: Měření mobilního zařízení před optimalizací

Metrika	Hlavní strana	Transportbeton	Ke stažení	Provozovny	Provozovna Český Krumlov
TTFB	426 ms	257 ms	387 ms	259 ms	392 ms
DOM Loading	434 ms	264 ms	397 ms	267 ms	399 ms
DOM Interactive	1117 ms	714 ms	855 ms	792 ms	927 ms
DOM Complete	1772 ms	813 ms	945 ms	1221 ms	1203 ms
Load Event	1773 ms	814 ms	946 ms	1223 ms	1204 ms
Speed Index	1137 ms	781 ms	921 ms	820 ms	909 ms
Start render	929 ms	679 ms	817 ms	663 ms	801 ms
First Meaningful Paint	976 ms	715 ms	856 ms	699 ms	845 ms
Time to First Interactive	9211 ms	3317 ms	4569 ms	4414 ms	4634 ms
Velikost HTML	40.35 KB	23.12 KB	41.64 KB	35.50 KB	24.64 KB
Velikost CSS	147.52 KB	114.49 KB	114.49 KB	115.35 KB	115.35 KB
Velikost fontů	18.03 KB	0.00 KB	0.00 KB	47.39 KB	47.39 KB
Velikost obrázků	589.03 KB	97.19 KB	104.00 KB	513.73 KB	261.64 KB
Kontakt v hlavičce	1034 ms	745 ms	912 ms	745 ms	878 ms
Logo	1034 ms	745 ms	912 ms	745 ms	878 ms
Zobrazení menu	1145 ms	745 ms	912 ms	745 ms	878 ms
Zobrazení nadpisu	1167 ms	745 ms	912 ms	745 ms	878 ms
Zobrazení tlačítka pro stažení	x	x	912 ms	x	x
Zobrazení služeb na HP	1167 ms	x	x	x	x

Tabulka 5.2: Měření stolního počítače před optimalizací

5.6 Optimalizace

5.6.1 Optimalizace technologií

Přechod z HTTP na HTTP/2

Jelikož webová prezentace nevyužívala protokol HTTP/2, v nastavení webhostingových služeb jsme zažádali o certifikát pro doménu a přepnutí serveru na protokol HTTP/2.

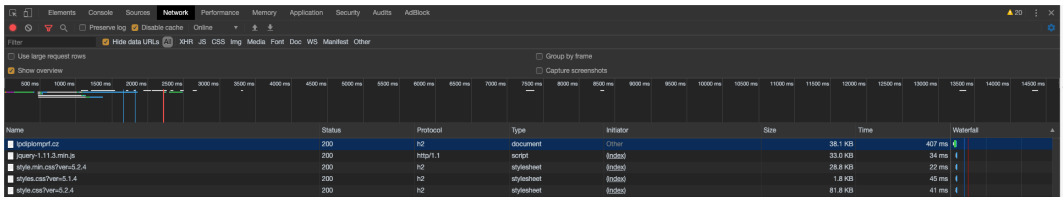
Aktualizace CMS a využitých rozšíření

Jak již bylo zmíněno, webová prezentace běží na CMS Wordpress a v rámci optimalizace technologií byla provedena jeho aktualizace na nejnovější verzi 5.3 a zároveň proběhla aktualizace všech používaných pluginů.

Aktualizace externích knihoven

Dále jsme zanalyzovali používané externí knihovny a zkontrolovali, zda využívají pro přenos protokol HTTP/2. V tomto kroku jsme zjistili, že web načítá knihovnu *jQuery* z externího zdroje přes HTTP viz obrázek 5.7. Nicméně je zde také vidět, že se jedná o duplikátní request, kterým jsme se zabývali v části optimalizace objemu dat.

Další externí knihovnou kterou web používá k zobrazování map je *Google Maps*. Tato knihovna používá pro přenos HTTP protokol. Nicméně hlavním problémem map bylo, že byly označeny vodoznakem použití pro vývojářské účely, jelikož využívaly API bez validního klíče. Po diskuzi s firmou jsme z ekonomických důvodů nahradili knihovnu *Google Maps* za *API Mapy.cz*. Zároveň jsme odstranili i plugin *Simple Map*, který knihovnu využíval.



Obrázek 5.7: Request na jQuery knihovnu přes HTTP

HTTP komprese

Webová prezentace nepoužívala kompresi souborů. Proto jsme aktivovali kompresi *Brotli* a přidali jsme do souboru *.htaccess* následující kód:

```
<IfModule mod_brotli.c>
  AddOutputFilterByType BROTLI_COMPRESS application/
     javascript
  AddOutputFilterByType BROTLI_COMPRESS application/json
  AddOutputFilterByType BROTLI_COMPRESS application/
     vnd.ms-fontobject
  AddOutputFilterByType BROTLI_COMPRESS application/xhtml
     +xml
  AddOutputFilterByType BROTLI_COMPRESS application/xml
  AddOutputFilterByType BROTLI_COMPRESS application/
     font-sfnt
  AddOutputFilterByType BROTLI_COMPRESS font/otf
  AddOutputFilterByType BROTLI_COMPRESS font/ttf
  AddOutputFilterByType BROTLI_COMPRESS image/svg+xml
  AddOutputFilterByType BROTLI_COMPRESS image/
     vnd.microsoft.icon
  AddOutputFilterByType BROTLI_COMPRESS text/plain
  AddOutputFilterByType BROTLI_COMPRESS text/css
  AddOutputFilterByType BROTLI_COMPRESS text/xml
  AddOutputFilterByType BROTLI_COMPRESS text/html
</IfModule>
```

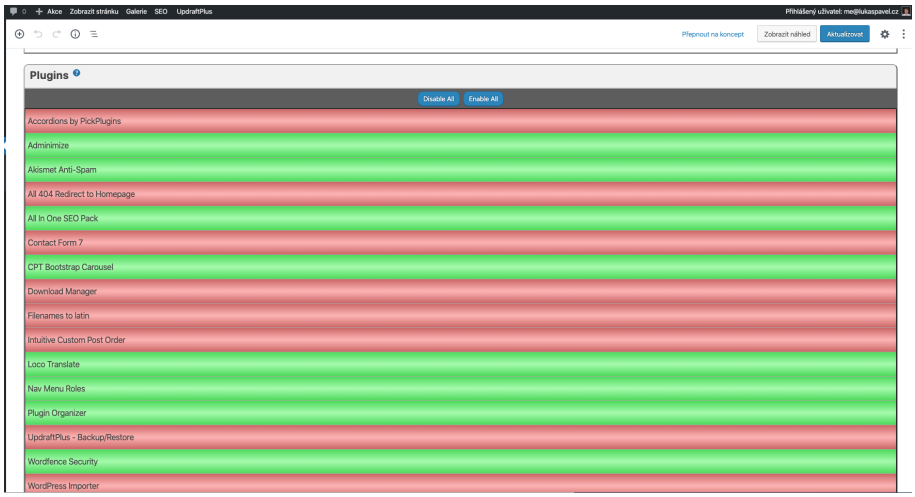
5.6.2 Optimalizace objemu dat

Nevyužité zdroje a duplikace

Provedli jsme analýzu requestů v nástroji *Chrome Dev Tools* a nalezli nepotřebné a duplicitní scripty a soubory. Například na všech stránkách se duplicitně načítala knihovna *jQuery*.

Wordpress pluginy velice často přidávají své CSS styly a scripty do hlavičky

na všechny stránky a není tomu jinak ani v našem případě. Proto jsme využili plugin *Plugin Organizer*, pomocí kterého jsme zakázali nepoužívané pluginy na konkrétních stránkách, jak ukazuje obrázek 5.8.



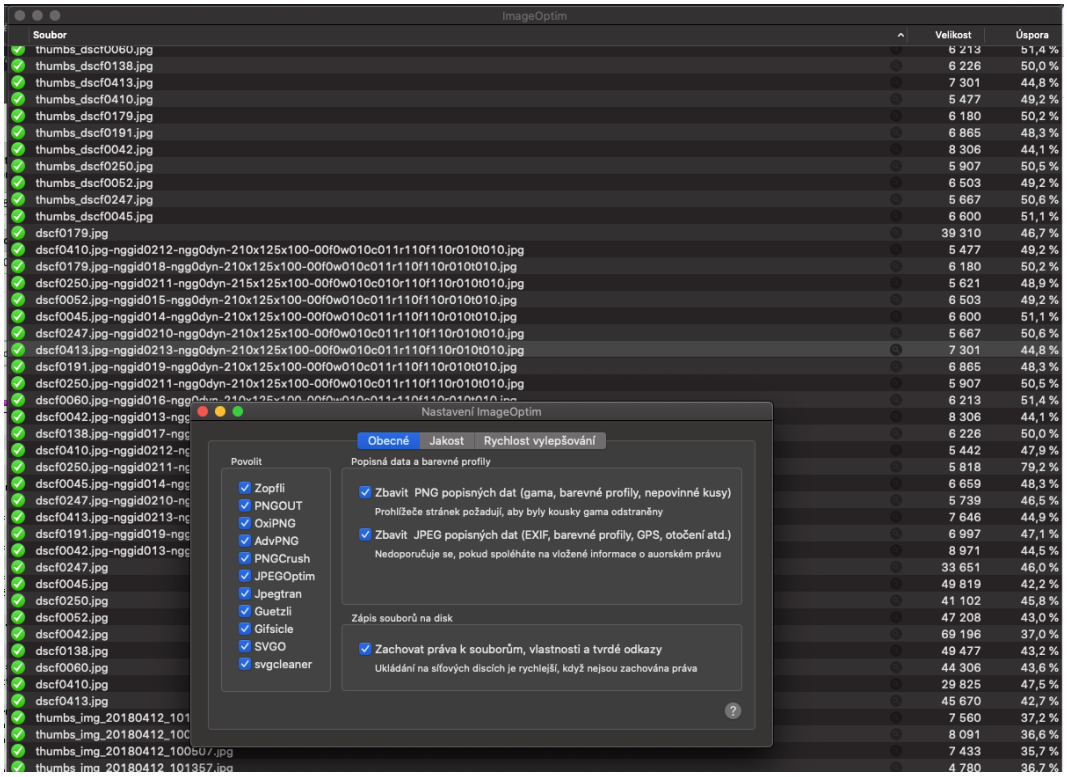
Obrázek 5.8: Použití pluginu Plugin Organizer

Optimalizace obrázků

Pro analýzu obrázků použitých na webu jsme využili nástroj *WebPageTest* a jeho test *Image Analysis* podle metodiky. Obrázky, které byly použité v šabloně pro CMS Wordpress, jsme stáhli a nahradili.

Vzhledem k tomu, že webová prezentace obsahuje i reference ve formě fotogalerie, pro optimalizaci obrázků jsme využili nástroj *ImageOptim*. Stáhli jsme zdrojové soubory, provedli optimalizaci a následně je nahradili přímo na serveru. Nastavení nástroje *ImageOptim* je patrné z obrázku 5.9 na další straně.

Pro optimalizaci nově přidaných obrázků do CMS Wordpress jsme využili plugin *Smush Image Compression and Optimization*, který provádí základní optimalizaci automaticky při nahrávání.



Obrázek 5.9: Použití nástroje ImageOptim

Optimalizace fontů

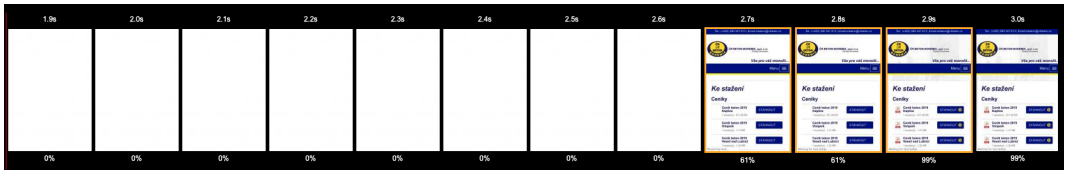
V rámci odstraňování nepotřebných knihoven a scriptů jsme odstranili i načítaný, ale nepoužívaný font *Font Awesome*. Ponechali jsme pouze font *Glyphicons Halflings* na hlavní straně, který je zde používaný pro carusel a v rámci optimalizace fontů bylo zkontrolováno, zda má dostupné všechny typy formátů a jeho forma načítání.

Minifikace HTML, CSS, JavaScript

Pro minifikaci zdrojových souborů v šabloně jsme využili nástroj *Gulp*. Pro ostatní zdroje, zejména pak pro zdroje z ostatních pluginů, jsme využili plugin *Hummingbird*, který najde všechny použité zdroje na webové prezentaci a dokáže provést jejich minifikaci.

5.6.3 Optimalizace času

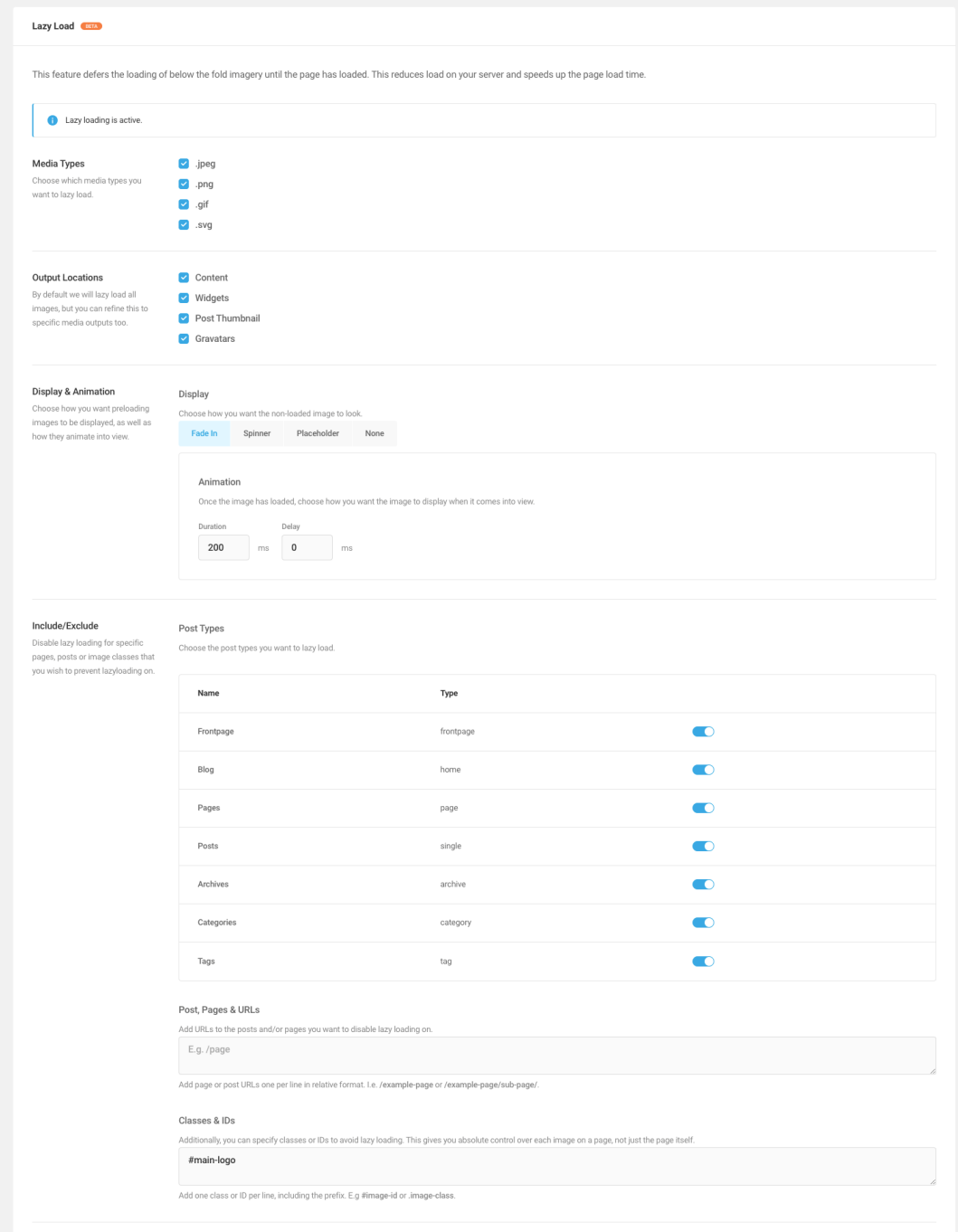
Při optimalizaci času jsme zanalyzovali *Filmstrip* v nástroji WebPageTest a zjistili, že dochází k začátku vykreslování stránky až po stažení všech zdrojů, jak ukazuje obrázek 5.10.



Obrázek 5.10: Filmstrip nástroje WebPageTest

LazyLoading

Pro LazyLoading obrázků jsme využili již používaný plugin *Smush Image Compression and Optimization* s následujícím nastavením, viz obrázek 5.11 na další straně. Na něm můžeme vidět, že jsme zakázali LazyLoading na logu, které je umístěné v hlavičce, neboť jedna z vlastních metrik měří právě čas jeho zobrazení.



Obrázek 5.11: Nastavení pluginu Smush Image Compression and Optimization

Async, Defer a kritické zdroje

Následně jsme provedli odložení načtení všech scriptů, kromě knihovny *jQuery*, pomocí atributu defer. Jediné *jQuery* zůstalo v kritické cestě, neboť je potřeba zajistit, aby bylo načteno před ostatními knihovnami, protože jsou na ni závislé.

Testovali jsme i využití atributu async, ale docházelo k pozdějšímu načtení než u ostatních knihoven. Pro přidání atributů defer ke scripům jsme využili následující filtr, který jsme přidali do souboru *functions.php* v šabloně CMS Wordpress:

```
function wsds_async_scripts( $tag, $handle, $src ) {
  $defer_scripts = array(
    'ajax',
    'smush-lazy-load',
    'script',
    'ngg_lightbox_context',
    'ngg_common',
    'photocrati-nextgen_basic_thumbnails',
    'simplelightbox-0',
    'simplelightbox-1',
    'frontjs',
    'jquery-choosen',
    'jquery-form'
  );
  if ( in_array( $handle, $defer_scripts ) ) {
    return '<script src="' . $src . '" defer="defer"
       type="text/javascript"></script>' . "\n";
  }
  return $tag;
}
add_filter( 'script_loader_tag', 'wsds_async_scripts',
   10, 3 );
```

Všechny styly a scripty jsme přesunuli do patičky webu, aby neblokovali načítání. K tomu jsme využili plugin *Hummingbird*, jelikož je schopný najít všechny styly a scripty napříč celým webem.

Po odstranění všech stylů z kritické cesty se uživateli načítalo nenastylované HTML. Proto jsme nechali vygenerovat kritické CSS styly pomocí scriptu, který využíval NPM balíček *CriticalCSS*. Ty jsme následně umístili do hlavičky celého webu.

Prefetching

Dalším krokem bylo nastavení, které zajistí, aby se prohlížeč při parsování HTML již začal připojovat na servery, ze kterých jsou nahrávány scripty třetích stran viz ukázka níže:

```
1 <link href="https://ajax.googleapis.com" rel="preconnect"
      crossorigin>
2 <link href="https://api.mapy.cz" rel="preconnect"
     crossorigin>
3 <link href="https://www.google-analytics.com" rel="
     preconnect" crossorigin>
4 <link href="https://mapserver.mapy.cz" rel="preconnect"
     crossorigin>
```

Zároveň jsme převedli logo společnosti do formátu *base64*, aby nedocházelo ke zpoždění v zobrazení loga v hlavičce.

Výsledkem optimalizace času je neblokované vykreslování jak na mobilním zařízení, tak na stolním počítači. Jak je patrné z obrázku níže.

Obrázek 5.12: Použití nástroje ImageOptim

5.6.4 Optimalizace serveru a zdrojů

Cache

V rámci optimalizace serveru a zdrojů jsme nastavili cache pomocí souboru *.htaccess*, kdy jsme nastavili expiraci všech stahovaných zdrojů na jeden měsíc, neboť zdrojové soubory pro daný web se mění jen výjimečně.

```
1 <FilesMatch "\.(js|css|jpg|gif|png|pdf|swf|svg|svgz|ico|
     ttf|ttc|otf|eot|woff|woff2|webp)$">
2  <IfModule mod_headers.c>
3     ExpiresActive On
```

```
4       ExpiresDefault  "access plus 1 month"
5       Header set Cache-Control "public, immutable,
            max-age=2628000, s-maxage=2628000"
6       Header set Access-Control-Allow-Origin "*"
7     </IfModule>
8   </FilesMatch>
```

CDN server

Posledním krokem v celé optimalizaci bylo přesunutí všech obrázků šablony, CSS stylů, scriptů na CDN server. K tomu jsme využili služeb *CDN77.com* a propojili ho s CMS Wordpress pomocí pluginu *WP Fastest Cache*. Následně jsme provedli následující nastavení v rámci administrace *CDN77*:

- Nastavili omezení na Českou republiku a Německo.
- Omezili dostupnost pouze na datacentra v blízkosti České republiky.

Následně jsme ověřili, že propojení na CDN server funguje a zdroje se stahují z jeho serveru. Taktéž jsme nastavili prefetching k připojení k serveru CDN a vložili link do hlavičky webu, viz ukázka kódu níže.

```
1 <link href="https://1406335708.rsc.cdn77.org" rel="
      preconnect" crossorigin>
```

5.7 Zaznamenání hodnot po optimalizaci

Metrika	Hlavní strana	Transportbeton	Ke stažení	Provozovny	Provozovna Český Krumlov
TTFB	945 ms	1103 ms	949 ms	814 ms	818 ms
DOM Loading	971 ms	1129 ms	975 ms	840 ms	841 ms
DOM Interactive	2340 ms	2278 ms	2205 ms	1971 ms	1972 ms
DOM Complete	2901 ms	2406 ms	2618 ms	2638 ms	2630 ms
Load Event	2918 ms	2411 ms	2622 ms	2651 ms	2646 ms
Speed Index	2157 ms	1754 ms	1745 ms	1514 ms	1702 ms
Start render	1204 ms	1316 ms	1281 ms	1015 ms	1040 ms
First Meaningful Paint	1232 ms	1338 ms	1312 ms	1037 ms	1064 ms
Time to First Interactive	2636 ms	2174 ms	2382 ms	2091 ms	2070 ms
Velikost HTML	11.63 KB	6.81 KB	10.35 KB	8.28 KB	7.26 KB
Velikost CSS	19.64 KB	13.87 KB	13.87 KB	17.67 KB	17.69 KB
Velikost fontů	18.03 KB	0.00 KB	0.00 KB	0.00 KB	0.00 KB
Velikost obrázků	233.02 KB	20.00 KB	22.25 KB	543.18 KB	201.12 KB
Kontakt v hlavičce	1223 ms	1334 ms	1300 ms	1012 ms	1034 ms
Logo	2023 ms	2267 ms	2223 ms	1712 ms	1745 ms
Zobrazení menu	2023 ms	2267 ms	2223 ms	1712 ms	1745 ms
Zobrazení nadpisu	2934 ms	2267 ms	2223 ms	2012 ms	2023 ms
Zobrazení tlačítka pro stažení	x	x	2523 ms	x	x
Zobrazení služeb na HP	2934 ms	x	x	x	x

Tabulka 5.3: Měření mobilního zařízení po optimalizaci

Metrika	Hlavní strana	Transportbeton	Ke stažení	Provozovny	Provozovna Český Krumlov
TTFB	208 ms	187 ms	202 ms	187 ms	196 ms
DOM Loading	215 ms	194 ms	210 ms	195 ms	204 ms
DOM Interactive	498 ms	428 ms	495 ms	443 ms	465 ms
DOM Complete	676 ms	490 ms	600 ms	590 ms	598 ms
Load Event	682 ms	491 ms	601 ms	595 ms	603 ms
Speed Index	645 ms	421 ms	496 ms	433 ms	481 ms
Start render	296 ms	266 ms	331 ms	269 ms	284 ms
First Meaningful Paint	317 ms	288 ms	367 ms	290 ms	309 ms
Time to First Interactive	2597 ms	2194 ms	2413 ms	2074 ms	2072 ms
Velikost HTML	11.63 KB	6.81 KB	10.35 KB	8.28 KB	7.26 KB
Velikost CSS	19.66 KB	13.87 KB	13.87 KB	17.69 KB	17.67 KB
Velikost fontů	18.03 KB	0.00 KB	0.00 KB	0.00 KB	0.00 KB
Velikost obrázků	196.17 KB	20.59 KB	22.84 KB	724.97 KB	257.96 KB
Kontakt v hlavičce	367 ms	312 ms	378 ms	312 ms	367 ms
Logo	489 ms	489 ms	500 ms	389 ms	389 ms
Zobrazení menu	812 ms	489 ms	500 ms	389 ms	389 ms
Zobrazení nadpisu	723 ms	489 ms	512 ms	512 ms	567 ms
Zobrazení tlačítka pro stažení	x	x	532 ms	x	x
Zobrazení služeb na HP	723 ms	x	x	x	x

Tabulka 5.4: Měření stolního počítače po optimalizaci

5.8 Vyhodnocení optimalizace

Metrika	Hlavní strana	Transportbeton	Ke stažení	Provozovny	Provozovna Český Krumlov	průměrně
TTFB	-23,53%	0,27%	-9,97%	-7,11%	-13,14%	-10,69%
DOM Loading	-22,76%	0,18%	-10,05%	-7,28%	-11,69%	-10,32%
DOM Interactive	22,80%	14,04%	12,22%	24,71%	31,93%	21,14%
DOM Complete	48,19%	30,64%	19,15%	44,37%	19,33%	32,33%
Load Event	47,91%	30,56%	19,07%	44,13%	18,88%	32,11%
Speed Index	26,48%	35,42%	33,52%	40,28%	32,75%	33,69%
Start render	50,29%	49,71%	49,05%	55,87%	56,39%	52,26%
First Meaningful Paint	50,42%	49,70%	48,99%	55,76%	56,25%	52,22%
Time to First Interactive	76,98%	35,53%	33,24%	54,61%	55,37%	51,15%
Velikost HTML	70,80%	70,54%	75,14%	76,68%	70,54%	72,74%
Velikost CSS	86,69%	87,89%	87,89%	84,69%	84,55%	86,34%
Velikost fontů	0%	0%	0%	100,00%	0%	x
Velikost obrázků	56,03%	79,06%	78,26%	27,04%	-80,41%	31,99%
Kontakt v hlavičce	51,08%	49,35%	49,36%	56,64%	57,13%	52,71%
Logo	19,08%	13,93%	13,40%	26,65%	27,65%	20,14%
Zobrazení menu	19,08%	13,93%	13,40%	26,65%	27,98%	20,21%
Zobrazení nadpisu	14,21%	14,29%	13,40%	13,80%	16,51%	14,44%
Zobrazení tlačítka pro stažení	x	x	8,25%	x	x	8,25%
Zobrazení služeb na HP	2,59%	x	x	x	x	2,59%
Celkové zrychlení						31,30%

Tabulka 5.5: Vyhodnocení měření mobilního zařízení

Na porovnání hodnot před a po optimalizaci u mobilního zařízení můžeme vidět, že v průměru došlo ke zrychlení o 30,04%. U metriky TTFB a DOM Loading můžeme vidět naopak zpomalení a to z důvodu přechodu na HTTP/2 a nutnosti zabezpečeného připojení. Dále můžeme vidět zvětšení datové velikosti obrázků na detailu provozovny a to z důvodu náhrady externí knihovny *Google Maps* za *API Mapy.cz* pro zobrazení mapy s bodem, kde se provozovna nachází. Tyto obrázky se nahrávají asynchroně a nemají tak vliv na rychlost vykreslování nebo načtení zbytku stránky. Velikost fontu záměrně nepočítáme do počítání celkového zrychlení, jelikož buď došlo k odstranění fontu ze stránky, nebo byl již optimalizován a zkresloval by tak celkové zrychlení.

Metrika	Hlavní strana	Transportbeton	Ke stažení	Provozovny	Provozovna Český Krumlov	průměrně
TTFB	51,17%	27,24%	47,80%	27,80%	50,00%	40,80%
DOM Loading	50,46%	26,52%	47,10%	26,97%	48,87%	39,98%
DOM Interactive	55,42%	40,06%	42,11%	44,07%	49,84%	46,30%
DOM Complete	61,85%	39,73%	36,51%	51,68%	50,29%	48,01%
Load Event	61,53%	39,68%	36,47%	51,35%	49,92%	47,79%
Speed Index	43,27%	46,09%	46,15%	47,20%	47,08%	45,96%
Start render	68,14%	60,82%	59,49%	59,43%	64,54%	62,48%
First Meaningful Paint	67,52%	59,72%	57,13%	58,51%	63,43%	61,26%
Time to First Interactive	71,81%	33,86%	47,19%	53,01%	55,29%	52,23%
Velikost HTML	71,18%	70,54%	75,14%	76,68%	70,54%	72,82%
Velikost CSS	86,67%	87,89%	87,89%	84,66%	84,68%	86,36%
Velikost fontů	0%	0%	0%	100,00%	100,00%	x
Velikost obrázků	66,70%	78,81%	78,04%	-41,12%	1,41%	36,77%
Kontakt v hlavičce	64,51%	58,12%	58,55%	58,12%	58,20%	59,50%
Logo	52,71%	34,36%	45,18%	47,79%	55,69%	47,15%
Zobrazení menu	29,08%	34,36%	45,18%	47,79%	55,69%	42,42%
Zobrazení nadpisu	38,05%	34,36%	43,86%	31,28%	35,42%	36,59%
Zobrazení tlačítka pro stažení	x	x	41,67%	x	x	41.67%
Zobrazení služeb na HP	38,05%	x	x	x	x	38.05%
Celkové zrychlení						51,65%

Tabulka 5.6: Vyhodnocení měření stolního počítače

Na porovnání hodnot před a po optimalizaci u stolního počítače můžeme vidět, že v průměru došlo ke zrychlení o 51,13%. Na stránce výpisu provozoven můžeme vidět zvýšení velikosti obrázků a opět stejně jako u mobilního zařízení je to z důvodu výměny knihovny pro zobrazování map. Změna se projevila až na stolním počítači z důvodu, že je velikost mapy na mobilním zařízení výrazně menší oproti stolnímu počítači, a tak stahuje méně obrázků. Zároveň můžeme vidět, že vlivem rychlejšího připojení a menší latence, již nedochází ke zpomalení u metrik TTFB a DOM Loading jako u mobilního zařízení.

Dále můžeme vidět v tabulkách 5.3 a 5.4, že metrika Load Event splňuje hlavní cíl definované metodiky na všech měřených stránkách a lze ji tak považovat za úspěšnou. Zároveň můžeme vidět, že došlo k výraznému zrychlení načtení u metriky "Kontakt v hlavičce" jakožto jednoho z hlavních cílů webové prezentace.

5.9 Nastavení procesu pro pravidelné sledování rychlosti načítání

Jelikož se jedná o dynamický web, byl nastaven firemní proces pro pravidelné měření rychlosti načítání webové prezentace jeden krát za tři měsíce vzhledem k frekvenci úprav.

Závěr

Hlavním cílem diplomové práce byl návrh metodiky pro optimalizaci webového frontendu z pohledu rychlosti načítání a následné prvotní ověření metodiky na webové prezentaci konkrétní firmy. Vlastní metodika vychází z provedené komentované rešerše na téma používaných metod optimalizace rychlosti načítání.

Pro splnění cílů bylo nejdříve potřeba najít firmu, na které byla ověřována vytvářená metodika. Poté byl vytvořen přehled technologií používaných pro návrh webového frontendu. Následovalo zpracování rešerše na téma metod optimalizace rychlosti načítání na webovém frontendu z různých hledisek (technologie, objem dat, čas, server a zdroj). Na základě provedené rešerše byla sestavena vlastní metodika a zvoleny vhodné nástroje pro testování a měření. Dále proběhlo prvotní ověření vytvořené metodiky na webové prezentaci firmy ČR Beton Bohemia spol. s r.o. Vlastní metodika je zaměřena na vyhodnocování vícenásobného měření a nastavení parametrů měření podle počáteční analýzy.

Začátek metodiky tvoří analýza klienta, ve které se zabýváme seznámením se s firmou a jejich současnou webovou prezentací. Spočívá v analýze podkladů, které má firma dostupné z vytváření webové prezentace, a dat o přístupu uživatelů k webové prezentaci. Zároveň obsahuje validaci nastavených cílů webové prezentace. Následuje analýza současných technologií využitých na webové prezentaci.

Poté se zaměřujeme na definici obecných a vlastních metrik, které budou následně měřeny. V této části je kladen důraz především na to, aby vlastní metriky vycházely z předešlé analýzy firmy a korespondovaly s cíli webové prezentace.

Dále se zabýváme způsobem měření rychlosti načítání webové prezentace a nástroji pro měření. Měření musí být provedeno zvlášť pro mobilní zařízení a zvlášť pro stolní počítač s odpovídající rychlostí připojení, latencí a rozlišením, které vychází z analýzy v první části. Dále je v této části metodiky kladen důraz na vyhodnocování metrik na základě určení středí hodnoty z vícenásobného měření. Zároveň měření musí probíhat na několika stránkách webu tak, aby pokrylo co největší část webu.

Následuje měření a zaznamenání definovaných metrik před optimalizací podle definovaného způsobu měření.

Další část metodiky se zabývá samotnou optimalizací rychlosti načítání webové prezentace. Nejdříve optimalizujeme technologie, neboť mají globální dopad. Následuje optimalizace velikosti objemu přenesených dat, poté optimalizace času a začátku vykreslování webové stránky. V poslední části optimalizace se zaměřujeme na zefektivnění odpovědi ze severu.

Po optimalizaci opět následuje měření a zaznamenání definovaných metrik stejným způsobem, jako před optimalizací.

Následně se zaměřujeme na porovnávání naměřených hodnot a jejich vyhodnocení.

V poslední části metodiky zdůrazňujeme potřebu nastavit proces pro pravidelné měření rychlosti načítání, protože pokud se jedná o dynamickou webovou prezentaci, její obsah a rychlost načítání se může měnit.

Po vytvoření metodiky proběhlo její prvotní ověření na webové prezentaci firmy ČR Beton Bohemia spol. s r.o. Z vyhodnocení optimalizace je patrné, že vytvořená metodika pro optimalizaci rychlosti načítání byla úspěšná a efektivní. U mobilního zařízení došlo k zrychlení webové prezentace v průměru o 31,30% a začátku vykreslování stránky o 52,26%. U stolního počítače došlo k zrychlení v průměru o 51,65% a začátku vykreslování stránky o 62,48%.

Seznam použité literatury

[1] MICHÁLEK, Martin. K čemu je dobrý Bootstrap a frontend frameworky?. Zdrojak.cz [online]. 2013, 6.12.2013 [cit. 2019-10-13]. Dostupné z: https://www.zdrojak.cz/clanky/k-cemu-je-dobry-bootstrap-frontend-frameworky/

[2] BENITTE, Raphaël, Sacha GREIF a Michael RAMBEAU. Stateofjs.com [online]. 2018 [cit. 2019-10-13]. Dostupné z: https://2018.stateofjs.com/

[3] JACKSON, Brian. CSS Preprocessors - Sass vs LESS [online]. 6.6.2017. 2017 [cit. 2019-10-13]. Dostupné z: https://www.keycdn.com/blog/sass-vs-less

[4] Hypertext Markup Language. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-10-13]. Dostupné z: https://cs.wikipedia.org/wiki/HyperText_Markup_Language

[5] KOSEK, Jiří. Úvod do historie. Htmlguru.cz [online]. [cit. 2019-10-13]. Dostupné z: http://htmlguru.cz/uvod-historie.html

[6] Web Platform WG specifications. W3.org [online]. [cit. 2019-10-13]. Dostupné z: https://www.w3.org/WebPlatform/WG/PubStatus

[7] Cascading Style Sheets. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-10-13]. Dostupné z: https://en.wikipedia.org/wiki/Cascading_Style_Sheets

[8] ATKINS JR., Tab, Elika J. ETEMAD a Florian RIVOAL, ed. CSS Snapshot 2017. W3.org [online]. 2017, 31.1.2017 [cit. 2019-10-13]. Dostupné z: https://www.w3.org/TR/css-2017/

[9] Javascript. MDN Web Docs [online]. 2005, 30.9.2019 [cit. 2019-10-13]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript

[10] SOUDERS, Steve. Evaluating rendering metrics: Speed Matters. SpeedCurce [online]. 11.12.2017 [cit. 2019-10-13]. Dostupné z: https://speedcurve.com/blog/rendering-metrics/

[11] MICHÁLEK, Martin. Nástroje pro analýzu rychlosti načtení stránky. Vzhurudolu.cz [online]. 27. 4. 2016 [cit. 2019-10-13]. Dostupné z: https://www.vzhurudolu.cz/prirucka/rychlost-nastroje

[12] State of the Web. Httparchive.org [online]. 2019 [cit. 2019-10-13]. Dostupné z: http://httparchive.org/trends.php?s=Top1000&minlabel=Nov+15+2010&maxlabel=Jan+1+2018#bytesJS&reqJS

[13] ARSENAULT, Cody. An Overview of the RAIL Performance Model. Keycdn.com [online]. 28.6.2018 [cit. 2019-10-13]. Dostupné z: https://www.keycdn.com/blog/rail-performance-model

[14] TILP, Dominik. Frontend Performance (když React a Webpack nejsou řešení) [online]. In: . 3.6.2017 [cit. 2019-10-13]. Dostupné z: https://www.superlectures.com/barcampbrno2017/frontend-performance-kdyz-react-a-webpack-nejsou-reseni

[15] HOFFMAN, Billy. Improving Search Rank by Optimizing Your Time to First Byte. In: Https://moz.com [online]. 26.9.2013 [cit. 2019-10-13]. Dostupné z: https://moz.com/blog/improving-search-rank-by-optimizing-your-time-to-first-byte

[16] SEXTON, Patrick. What is Document.readyState?. In: Varvy.com [online]. 12.8.2015 [cit. 2019-10-13]. Dostupné z: https://varvy.com/performance/document-ready-state.html

[17] SEXTON, Patrick. What is domLoading?. In: Varvy.com [online]. 1.8.2015 [cit. 2019-10-13]. Dostupné z: https://varvy.com/performance/domloading.html

[18] SEXTON, Patrick. What is domInteractive?. In: Varvy.com [online]. 10.8.2015 [cit. 2019-10-13]. Dostupné z: https://varvy.com/performance/dominteractive.html

[19] SEXTON, Patrick. What is domContentLoaded?. In: Varvy.com [online]. 6.9.2015 [cit. 2019-10-13]. Dostupné z: https://varvy.com/performance/domcontentloaded.html

[20] SEXTON, Patrick. What is domComplete?. In: Varvy.com [online]. 11.9.2015 [cit. 2019-10-13]. Dostupné z: https://varvy.com/performance/domcomplete.html

[21] EVERTS, Tammy. Images Are King: An Image Optimization Checklist for Everyone in Your Organization. In: Calendar.perfplanet.com [online]. 10.12.2014 [cit. 2019-10-13]. Dostupné z: https://calendar.perfplanet.com/2014/images-are-king-an-image-optimization-checklist-for-everyone-in-your-organization/

[22] MICHÁLEK, Martin. Lazy loading v kontextu webového frontendu: co to je a proč to dělat?. Vzhurudolu.cz [online]. 12. 8. 2019 [cit. 2019-10-13]. Dostupné z: https://www.vzhurudolu.cz/prirucka/lazy-loading

[23] HUME, Dean. Understanding Critical CSS. Preslav.me [online]. 13.8.2015 [cit. 2019-10-13]. Dostupné z: https://www.smashingmagazine.com/2015/08/understanding-critical-css/

[24] GRIGORIK, Ilya. Image Optimization. In: Developers.google.com: Web Fundamentals [online]. [cit. 2019-10-13]. Dostupné z: https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/image-optimization

[25] CIRKLOVÁ, Barbora. Rozdíl mezi rastrovou a vektorovou grafikou. In: Wplama.cz [online]. 27.9.2014 [cit. 2019-10-13]. Dostupné z: https://www.wplama.cz/rozdil-mezi-rastrovou-a-vektorovou-grafikou/

[26] OSMANI, Addy. Essential Image Optimization. In: Images.guide [online]. 7.10.2018 [cit. 2019-10-13]. Dostupné z: https://images.guide/

[27] BAR, Tomer. Faster Photos in Facebook for iOS. In: Facebook Engineering [online]. 28.1.2015 [cit. 2019-10-13]. Dostupné z: https://code.facebook.com/posts/857662304298232/faster-photos-in-facebook-for-ios/

[28] Hypertext Transfer Protocol. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-10-13]. Dostupné z: https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

[29] WALLS, Colin. Embedded Software: The Works. [s.l.]: Elsevier, 2006. 390 s. Dostupné online. ISBN 0750679549, 9780750679541. S. 344.

[30] JACKSON, Brian. What is the Difference Between HTTP and HTTPS?. Keycdn.com [online]. 21.9.2016 [cit. 2019-10-13]. Dostupné z: https://www.keycdn.com/blog/difference-between-http-and-https/

[31] GRIGORIK, Ilya. High-performance browser networking. Sebastopol, CA: O'Reilly, 2013. ISBN 978-1449344764. Dostupné z: https://hpbn.co/transport-layer-security-tls/#tls-handshake

[32] MICHÁLEK, Martin. WebP obrázky: datově úsporná alternativa k JPEG, PNG i GIF. Vzhurudolu.cz [online]. 10. 6. 2018 [cit. 2019-10-13]. Dostupné z: https://www.vzhurudolu.cz/prirucka/webp

[33] A new image format for the Web. Developers.google.com [online]. [cit. 2019-10-13]. Dostupné z: https://developers.google.com/speed/webp/

[34] Encyclopedia of Graphics File Formats, 2nd Edition. by Murray, James D. , Van Ryper, William ISBN: 1-56592-161-5

[35] PATEL, Priyesh. What exactly is Node.js?. Freecodecamp.org [online]. 18.4.2018 [cit. 2019-10-13]. Dostupné z: https://medium.freecodecamp.org/what-exactly-is-node-js-ae36e97449f5

[36] PURDILA, Adi. What Is Gulp?. Webdesign.tutsplus.com [online]. 31.1.2018 [cit. 2019-10-13]. Dostupné z: https://webdesign.tutsplus.com/tutorials/what-is-gulp--cms-30432

[37] MICHÁLEK, Martin. Atributy responzivních obrázků: srcset a sizes. Vzhurudolu.cz [online]. 23. 5. 2017 [cit. 2019-10-13]. Dostupné z: https://www.vzhurudolu.cz/prirucka/srcset-sizes

[38] MICHÁLEK, Martin. Picture, nová značka pro vkládání obrázků. Vzhurudolu.cz [online]. 21. 6. 2017 [cit. 2019-10-13]. Dostupné z: https://www.vzhurudolu.cz/prirucka/picture https://www.vzhurudolu.cz/prirucka/picture

[39] BASQUES, Kayce. Get Started With Analyzing Runtime Performance. Developers.google.com [online]. [cit. 2019-10-13]. Dostupné z: https://developers.google.com/web/fundamentals/performance/rail

[40] ARCHIBALD, Jake. SVGOMG! [online]. 5. 2. 2015 [cit. 2019-10-13]. Dostupné z: https://github.com/jakearchibald/svgomg/blob/master/README.md

[41] DINCULESCU, Monica, Rob DODSON, Jeff POSNICK a Ilya GRIGORIK. Web Font Optimization. Developers.google.com: Web Fundamentals [online]. [cit. 2019-10-13]. Dostupné z:https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/webfont-optimization

[42] STEIN, Bram. WEBFONT HANDBOOK. 1. 2017. ISBN 978-1-937557-64-5.

[43] KEARNEY, Meggin, Addy OSMANI, Kayce BASQUES a Jason MILLER. Measure Performance with the RAIL Model. Developers.google.com [online]. [cit. 2019-10-13]. Dostupné z: https://developers.google.com/web/fundamentals/performance/rail

[44] Content delivery network. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-10-13]. Dostupné z: https://cs.wikipedia.org/wiki/Content_delivery_network

[45] ORGEL, Elie. Setting up a Content Delivery Network (CDN) on Magento. Shimonsandler.com [online]. 19.4.2014 [cit. 2019-10-13]. Dostupné z: http://www.shimonsandler.com/cdn-with-magento/

[46] Remove Render-Blocking JavaScript. Developers.google.com [online]. [cit. 2019-10-13]. Dostupné z: https://developers.google.com/speed/docs/insights/BlockingJS

[47] What is HTTP/2?. Http2.github.io [online]. [cit. 2019-10-13]. Dostupné z: https://http2.github.io/

[48] HTTP/2. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-10-13]. Dostupné z: https://cs.wikipedia.org/wiki/HTTP/2

[49] STENBERG, Daniel. Http2 explained [online]. 11.11.2015 [cit. 2019-10-13]. Dostupné z: https://github.com/bagder/http2-explained/blob/master/en/part6.md

[50] MICHÁLEK, Martin. Rychlý protokol HTTP/2: S nasazením na weby na nic nečekejte. Vzhurudolu.cz [online]. 1. 1. 2019 [cit. 2019-10-13]. Dostupné z: https://www.vzhurudolu.cz/prirucka/http-2

[51] RENDLE, Robin. Prefetching, preloading, prebrowsing. Css-tricks.com [online]. 31.1.2016 [cit. 2019-10-13]. Dostupné z: https://css-tricks.com/prefetching-preloading-prebrowsing/

[52] ARSENAULT, Cody. Front End Optimization - 9 Tips to Improve Web Performance. Keycdn.com [online]. 10.4.2017 [cit. 2019-10-13]. Dostupné z: https://www.keycdn.com/blog/front-end-optimization

[53] COYIER, Chris. Active Gzip Compression. Css-tricks.com [online]. 30.6.2015 [cit. 2019-10-13]. Dostupné z: https://css-tricks.com/snippets/htaccess/active-gzip-compression/

[54] Can gzip Compression Really Improve Web Performance?. Royal.pingdom.com [online]. 15.8.2018 [cit. 2019-10-13]. Dostupné z: https://royal.pingdom.com/can-gzip-compression-really-improve-web-performance/

[55] Cache Definition and Explanation: Support. Keycdn.com [online]. 4.10.2018 [cit. 2019-10-13]. Dostupné z: https://www.keycdn.com/support/cache-definition-explanation

[56] ARSENAULT, Cody. HTTP Cache Headers - A Complete Guide. Keycdn.com [online]. 24.4.2018 [cit. 2019-10-13]. Dostupné z: https://www.keycdn.com/blog/http-cache-headers

[57] RACHEV, Preslav. Gulp vs Grunt. Why one? Why the Other?. Preslav.me [online]. 6.1.2015 [cit. 2019-10-13]. Dostupné z: https://preslav.me/2015/01/06/gulp-vs-grunt-why-one-why-the-other/

[58] About WebPagetest.org. Webpagetest.org [online]. [cit. 2019-10-13]. Dostupné z: https://https://www.webpagetest.org/about

[59] EVERTS, Tammy. Getting started with web performance? Here's what you need to focus on. Speedcurve.com [online]. [cit. 2019-10-13]. Dostupné z: https://speedcurve.com/blog/getting-started-web-performance/

[60] Naměřené rychlosti internetu na DSL.cz v lednu 2019. Dsl.cz [online]. 12.2.2019 [cit. 2019-10-13]. Dostupné z: http://www.dsl.cz/clanky/namerene-rychlosti-internetu-na-dsl-cz-v-lednu-2019

[61] ŽÁČEK, Josef. What's the difference between async vs defer attributes. Josef Zacek Web Development [online]. 2018, 8.3.2018 [cit. 2019-11-24]. Dostupné z: https://www.josefzacek.cz/blog/whats-the-difference-between-async-vs-defer-attributes/

[62] GRIGORIK, Ilya. Web Font Optimization [online]. [cit. 2019-11-24]. Dostupné z: https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/webfont-optimization

[63] MICHÁLEK, Martin. Webpack: Úplné základy a tutoriál k tomu. Vzhurudolu.cz [online]. 2019, 9.10.2019 [cit. 2019-11-24]. Dostupné z: https://www.vzhurudolu.cz/prirucka/webpack

[64] GRIFFIN, Jonathan. HTTP/2 vs HTTP/1: Everything you need to know about HTTP/2 and how it differs from HTTP/1. If you are not yet using HTTP/2, you will want to after reading this guide. Thewebmaster.com [online]. 2019, 3.2.2019 [cit. 2019-11-24]. Dostupné z: https://www.thewebmaster.com/hosting/2015/dec/14/what-is-http2-and-how-does-it-compare-to-http1-1/

[65] DAITYARI, Shaumik. Angular vs React vs Vue: Which Framework to Choose in 2019. Codeinwp.com [online]. 2019, 13.1.2019 [cit. 2019-12-02]. Dostupné z: https://www.codeinwp.com/blog/angular-vs-vue-vs-react/

[66] CALVANO, Paul. Brotli Compression – How Much Will It Reduce Your Content? Paulcalvano.com [online]. 2018, 25.7.2018 [cit. 2019-12-02]. Dostupné z: https://paulcalvano.com/index.php/2018/07/25/brotli-compression-how-much-will-it-reduce-your-content/

[67] MACCANA, Mike. You can't use Brotli for dynamic content. Certsimple.com [online]. 2017, 4.4.2017 [cit. 2019-12-02]. Dostupné z: https://certsimple.com/blog/nginx-brotli

[68] SOUDERS, Steve. Prebrowsing. Stevesouders.com [online]. 2013, 7.11.2013 [cit. 2019-12-02]. Dostupné z: http://www.stevesouders.com/blog/2013/11/07/prebrowsing/

[69] MARINI, Joe. Measuring Mobile Site Performance Using the W3C Navigation Timing API. Blogs.windows.com [online]. 2011, 2.8.2011 [cit. 2019-12-02]. Dostupné z: https://blogs.windows.com/windowsdeveloper/2011/08/02/measuring-mobile-site-performance-using-the-w3c-navigation-timing-api/

[70] MORENO, Nicolás Peña a Tim DRESSER. Element Timing API: Draft Community Group Report, 18 November 2019. Wicg.github.io [online]. 2019, 18.11.2019 [cit. 2019-12-02]. Dostupné z: https://wicg.github.io/element-timing/

[71] CHAMPAGNE, Jason. How to Minify Your Website's CSS, HTML & Javascript. Elegantthemes.com [online]. 2019, 17.1.2019 [cit. 2019-12-07]. Dostupné z: https://www.elegantthemes.com/blog/tips-tricks/how-to-minify-your-websites-css-html-javascript

Seznam tabulek

1. Tabulka 1.1: Základní porovnání Angular, React a Vue.js podle Shaumika Daityari
2. Tabulka 3.1: Způsob vnímání rychlosti načítání webové aplikace podle RAIL
3. Tabulka 3.2: Porovnání výhod a nevýhod výše zmíněných nástrojů
4. Tabulka 5.1: Měření mobilního zařízení před optimalizací
5. Tabulka 5.2: Měření stolního počítače před optimalizací
6. Tabulka 5.3: Měření mobilního zařízení po optimalizací
7. Tabulka 5.4: Měření stolního počítače po optimalizací
8. Tabulka 5.5: Vyhodnocení měření mobilního zařízení
9. Tabulka 5.6: Vyhodnocení měření stolního počítače

Seznam použitých zkratek

1. HTML - HyperText Markup Language
2. CSS - Cascading Style Sheets
3. JS - JavaScript
4. HTTP - Hypertext Transfer Protocol
5. HTTPs - Hypertext Transfer Protocol Secure
6. CDN - Content Delivery Network
7. SGML - Standard Generalized Markup Language
8. XHTML - Extensible Hypertext Markup Language
9. W3C - World Wide Web Consortium
10. XML - Extensible Markup Language
11. ECMA - European Computer Manufacturers Association
12. DOM - Document Object Model
13. RUM - Real user monitoring
14. API - Application Programming Interface
15. TTFB - Time to first byte
16. CSSOM - CSS Object Model
17. MIME - Multipurpose Internet Mail Extensions
18. TCP - Transmission Control Protocol
19. RFC - Request for Comments
20. ISO/OSI - Open System Interconnection model

21. TLS - Transport Layer Security
22. SSL - Secure Sockets Layer
23. QUIC - Quick UDP Internet Connections
24. UDP - User Datagram Protocol
25. JPEG - Joint Photographic Experts Group
26. MNG - Multiple-image Network Graphics
27. APNG - Animated Portable Network Graphics
28. PNG - Portable Network Graphics
29. GIF - Graphics Interchange Format
30. LZW - Lempel-Ziv-Welch 84
31. SVG - Scalable Vector Graphics
32. EOT - Embedded Open Type
33. TTF - TrueType Font
34. WOFF - Web Open Font Format
35. CPU - Central Processing Unit
36. DNS - Domain Name System
37. FPS - Frames Per Second
38. URL - Uniform Resource Locator
39. BSD - Berkeley Software Distribution
40. PHP - Hypertext Preprocessor
41. CMS - Content Management System
42. HAR - HTTP Archive format