Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Diplomová práce

Vliv CSS na výkon a rychlost načítání webových stránek

Jiří Kaštánek

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Jiří Kaštánek

Systémové inženýrství a informatika
Informatika

Název práce

Vliv CSS na výkon a rychlost načítání webových stránek

Název anglicky

The impact of CSS on web page loading speed and performance

Cíle práce

Tato diplomová práce se bude zabývat základním principem vykreslení webové stránky na straně klienta. Hlavním cílem je srovnání několika systémů a přístupů organizace CSS, v návaznosti na rychlost načtení webových stránek, v různých internetových prohlížečích na straně klienta.
Mezi dílčí cíle této práce patří:
- Vytvoření vzorových webových stránek pomocí různých přístupů organizace CSS, následné měření rychlosti načtení a vytížení na straně klienta pomocí vhodně zvolených nástrojů a metod.
- Porovnání rychlosti načtení jednoduchých oproti tzv. drahým CSS vlastnostem.
- Doporučení pro optimalizací webové stránky tak, aby výsledné načítání obsahu stránky bylo co nejrychlejší a nejplynulejší.

Metodika

Po představení teoretických východisek bude tato práce řešit problematiku možných nástrojů pro měření rychlosti načtení, ať už na straně prohlížeče či přes další alternativní nástroje. Například přes Webpagetest.org, Google PageSpeed Insight nebo záložku vývojáře v prohlížeči Chrome. Přes tyto nástroje bude provedeno měření průběhu načítání a odhalení prvků které jej brzdí, dále také bude sledována vytíženost počítače. Jednotlivá měření budou vždy testovat jeden princip organizace CSS. Po vykonaném testování a získaní výstupních dat, budou nadále tato data uspořádána a vyhodnocena.

Závěr bude formulován na základě výsledků získaných v praktické části práce.

Doporučený rozsah práce

60 – 70 stran

Klíčová slova

vykreslovací jádro, CSS, rychlost načtení webu, kritické CSS

Doporučené zdroje informací

BROWN, Tiffany B. CSS Master. Austrálie: Sitepoint, 2015. ISBN 978-0-9941826-2-3.

BUDD, Andy, Cameron MOLL a Simon COLLISON. CSS mastery: advanced web standards solutions. 2nd. ed. New York: Distributed to the Book trade in the United States by Springer-Verlag New York, 2009. ISBN 978-1-4302-2397-9.

MICHÁLEK, Martin. Vzhůru do (responzivního) webdesignu. Verze 1.1. Praha: vlastním nákladem autora, 2017. ISBN 978-80-88253-00-6.

PODJARNY, Guy. Responsive & Fast: Implementing High-Performance Responsive Design. Sebastopol, CA: O'Reilly Media, c2014. ISBN 978-1491911617.

SOUDERS, Steve. High performance web sites: essential knowledge for frontend engineers. Farnham: O'Reilly, c2007. ISBN 978-0596529307.

WAGNER, Jeremy L. a Ethan MARCOTTE. Web performance in action: building fast web pages. Shelter Island, NY: Manning Publications Co., [2017]. ISBN 978-1617293771.

Předběžný termín obhajoby

2019/20 LS – PEF

Vedoucí práce

Ing. Jan Masner, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 26. 8. 2019

Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 14. 10. 2019

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 04. 04. 2020

Čestné prohlášení

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

V Praze dne 5.4.2020 _____

Poděkování

Rád(a) bych touto cestou poděkoval(a) Ing. Jan Masner, Ph.D. za aktivní přístup, cenné rady a pomoc při tvorbě této práce. Dále bych chtěl poděkovat mé přítelkyni a rodině, za veškerou podporu, která se mi z jejich strany dostala.

Vliv CSS na výkon a rychlost načítání webových stránek

Abstrakt

Cílem této diplomové práce bylo zjistit, jakým způsobem ovlivňují různé typy organizace kaskádových stylů rychlost načtení a vykreslení webové stránky.

Pro zjištění přesných výsledků bylo použito měření rychlosti načtení a vykreslení na straně klienta, za pomoci vývojářského nástroje, jako je webový prohlížeč Google Chrome. Následně došlo k zaměření pomocí nástrojů Webpagetest.org a PageSpeed Insights.

Provedením měření pomocí výše uvedených nástrojů, došlo k interpretaci výsledků, na základě kterých lze říci, jaký typ organizace kaskádových stylů ve stránce je nejvhodnější pro zrychlení zobrazení webové stránky.

Jednoznačně došlo k určení nejrychleji načítané organizace kaskádových stylů. Dále také došlo k identifikaci určité hranice – neboli počtu použití CSS vlastností, od kterých dochází k prodlevě vykreslení stránky.

Klíčová slova: vykreslovací jádro, CSS, rychlost načtení webu, kritické CSS

The impact of CSS on web page loading speed and performance

Abstract

The aim of this diploma thesis was to find out how different types of organization of cascading styles affect the speed of loading and rendering web page.

Client-side load and render rates were used to determine accurate results, using a developer tool such as Google Chrome. Subsequently, it was focused on using Webpagetest.org and PageSpeed Insights.

By measuring with the above tools, the results were interpreted based on what type of organization of the cascading style sheets in a page is best suited to speed up a website.

Conclusively, the fastest-loaded cascading style organization was identified. In addition, a certain threshold has been identified – the number that CSS features have been used, from which page rendering is delayed.

Keywords: browser rendering engine, CSS, web load time, critical CSS

Obsah

Seznam obrázků

Seznam tabulek

Seznam použitých zkratek

CSS – Cascading Styl Sheets – Kaskádové styly

HTTP – Hypertext Transfer Protocol – Protokol sloužící pro přenos hypertextových dokumentů

HTTPS – Hypertext Transfer Protocol Secure – Protokol využívající zabezpečenou komunikaci

TLS – Transport Layer Security – Kryptografický protokol

URL – Uniform Resource Locator – Jednoznačné určení zdroje

DNS – Domain Name System – Systém pro překlad doménových jmen

JS – JavaScript – Skriptovací jazyk

XML – Extensible Markup Language – Značkovací jazyk

HTML – Hypertext Markup Language – Značkovací jazyk

SEO – Search Engine Optimization – Optimalizace pro vyhledávače

DOM – Document Object Model – Objektový model dokumentu

PSI – PageSpeed Insigths – Nástroj pro syntetické měření počítačem

CDN – Content Delivery Network – Síť pro doručování obsahu

CSSOM – CSS Object Model – CSS objektový model

FPS – Frames per second – Snímková frekvence

PHP – Hypertext Preprocessor – Skriptovací programovací jazyk

MB – MegaByte – Jednotka množství dat

RAM – Random Access Memory – Operační paměť

GB – Gigabyte – Jednotka množství dat

CPU – Central Processing Unit – Centrální procesorová jednotka

GPU – Graphics Processing Unit – Grafický procesor

PNG – Portable Network Graphics – Grafický formát pro rastrovou grafiku

JPG – Joint Photographic Experts Group – Grafický formát pro rastrovou grafiku

1 Úvod

Svět internetu ovlivňuje téměř veškerou lidskou populaci. Mnoho lidí na síti vyhledává zábavu, sociální kontakt, hledá informace či naopak prezentuje sebe či svou společnost. Je to též nástroj pro vytváření zisků. Kvalitu stránky ukazuje její návštěvnost a oblíbenost u uživatelů. To, jakým způsobem se uživateli zobrazuje, je klíčovou vlastností.

Provázaností, spokojeností a celkovou interakcí uživatele s konkrétní webovou stránkou se zabývají vývojáři zaměřující se na "user experience", tedy uživatelskou zkušenost. Web musí být přehledný, musí poskytovat ty nejdůležitější informace, kvůli kterým je člověk hledá a primárně musí být funkční.

Dle světových studií tráví průměrný člověk na internetu několik hodin denně. Za tento časový interval stihne mnoho úkonů. Obzvláště u stránek, které navštíví poprvé, je pro jeho spokojenost rozhodující doba, za jakou se celá stránka zobrazí do své plné podoby, a za jak dlouho z ní získá potřebné informace. Vykreslení v rámci milisekund mozek vyhodnotí jako okamžitou reakci, vteřinové odezvy vnímá s lehkou prodlevou, ale považuje je za přijatelné. Po zhruba deseti vteřinách ztrácíme zájem a hledáme jiné zdroje informací, případně jiné webové stránky.

Tento fakt si uvědomuje mnoho internetových prodejců na celém světě. Zrychlení načtení webové stránky zvyšuje konverzní poměr, může ovlivnit počet zobrazených stránek i spokojenost návštěvníků na stránkách a jeho opakovaný návrat. Spokojený zákazník znamená vyšší tržby. Proto se vývojáři webu ale i další vývojáři, co se zabývají pouze uživatelskou přívětivostí, začali zajímat, jakým způsobem lze načtení a s tím související vykreslování stránek ovlivnit.

Z tohoto důvodu se tato práce zaměřuje na porovnání rychlosti načtení a vykreslení tří typů organizace kaskádových stylů v menších webových stránkách tak, aby šlo určit, která z organizací (CSS stylů) má nejkratší čas načtení a vykreslení, a následně dokázala odborníkům napomoci při optimalizování a zrychlování webových stránek.

2 Cíl práce a metodika

2.1 Cíl práce

Tato diplomová práce byla zaměřena na základní princip vykreslení webové stránky na straně klienta. Hlavním cílem je srovnání několika systémů a přístupů organizace CSS, v návaznosti na rychlost načtení webových stránek, v různých internetových prohlížečích na straně klienta.

Mezi dílčí cíle této práce patří:

- Vytvoření vzorových webových stránek pomocí různých přístupů organizace CSS, následné měření rychlosti načtení a vytížení na straně klienta pomocí vhodně zvolených nástrojů a metod.
- Porovnání rychlosti načtení jednoduchých oproti tzv. drahým CSS vlastnostem.
- Doporučení pro optimalizací webové stránky tak, aby výsledné načítání obsahu stránky bylo co nejrychlejší a nejplynulejší.

2.2 Metodika

Po představení teoretických východisek bude tato práce řešit problematiku možných nástrojů pro měření rychlosti načtení, ať už na straně prohlížeče či přes další alternativní nástroje. Například přes Webpagetest.org, Google PageSpeed Insight nebo záložku vývojáře v prohlížeči Chrome. Přes tyto nástroje bude provedeno měření průběhu načítání a odhalení prvků které jej brzdí, dále také bude sledována vytíženost počítače. Jednotlivá měření budou vždy testovat jeden princip organizace CSS. Po vykonaném testování a získaní výstupních dat, budou nadále tato data uspořádána a vyhodnocena. Závěr bude formulován na základě výsledků získaných v praktické části práce.

3 Teoretická východiska

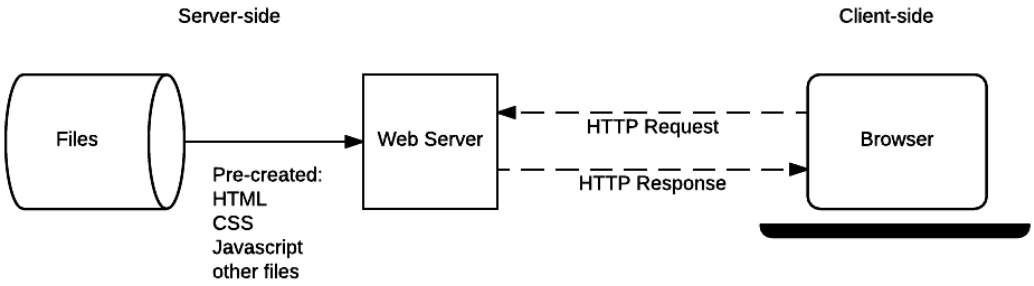
3.1 Načítání webové stránky

Načtení webové stránky je proces, který lze rozdělit na dvě části. První z nich je stažení dat – neboli komunikace webového prohlížeče a webového serveru pomocí protokolu HTTP. Druhá část procesu načtení webové stránky je samotné vykreslení, které probíhá na straně klienta, přímo ve webovém prohlížeči. [1]

Mezi nejčastější aspekty, které zpomalují načtení webové stránky patří velikost souborů, rychlost internetového připojení či nedostatečná optimalizace webových stránek. Charakteristika a přesnější průběh jednotlivých částí je uveden dále v textu. [2]

3.1.1 Stažení dat

Při tomto procesu je využíván protokol HTTP. Lze říci, že stažení dat do prohlížeče probíhá pomocí výpočetního modelu klient / server. Jak lze vidět na obrázku č. 1, proces stažení probíhá mezi klientem – tedy prohlížečem a webovým serverem. [1], [3]



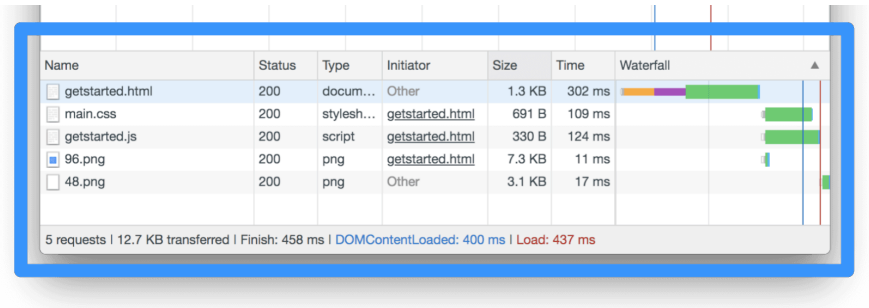
Obrázek 1 - Výpočetní model klient – server [3]

Postupem času je protokol HTTP nahrazován zabezpečenou verzí – HTTPS, která kombinuje HTTP společně s TLS nebo SSL a komunikace se stává bezpečnou. Uživatel ve svém prohlížeči pozná zabezpečenou stránku tak, že se v URL řádku prohlížeče zobrazí vlevo nahoře ikonka zámečku a začátek zadané adresy (HTTPS) obvykle zezelená. Tento systém v dnešní době používá čím dál větší množství webových stránek. V minulosti s touto technologií pracovaly hlavně stránky, pracující s citlivými daty, jako jsou banky, mailové servery, internetové obchody apod. [3]

Po zadání URL (tj. v překladu jedinečný identifikátor umístění) do prohlížeče proběhne překlad URL pomocí DNS serverů, následně je dotázán webový server a prohlížeč začne stahovat zdrojový soubor jako HTTP objekt. Tento proces probíhá jako

dotaz (tj. z angl. request) a odpověď (tj. z angl. response). Dotaz je zaslán na příslušný server, následně je vrácena odpověď, v které se nachází nejen výsledek, ale i samotný dotazovaný dokument. [2]

Kromě dokumentu se zdrojovým kódem stránky, se stahují také obrázky, externí JavaScriptové soubory, externí soubory CSS stylů, iframe, fonty a jiné HTTP objekty. Na některé tyto objekty se musí čekat s vykreslením stránky v prohlížeči, jako jsou například externí skripty a styly CSS. [2]



Obrázek 2 - Seznam stahovaných souborů, většina webových prohlížečů umožňuje tento seznam dopodrobna prohlížet a zkoumat [4]

Jelikož načítání musí proběhnout velmi rychle, prohlížeč je uzpůsoben k tomu, aby jeho HTTP objekty stahoval po několika najednou. Ovšem má to svá omezení, a proto není také dobré do hlavičky připojovat více externích stylů či skriptů. Tento proces se nazývá paralelní načítání. Když už je stažena hlavní stránka, prohlížeč ví, jaké HTTP objekty musí ještě stáhnout (například obrázky). Proto hned jak je to možné, pošle prohlížeč HTTP dotaz na jejich stažení. [4], [5]

3.1.2 Vykreslení

Vykreslení neboli renderování je proces, při kterém je ze stažených zdrojových souborů za pomoci vykreslovacího jádra prohlížeče složena a vykreslena stránka. Při načítání stránky je známo několik faktorů, které mohou zpomalovat jeho průběh.

Velikost souborů i v dnešní době rychlého připojení k internetu, hraje velmi důležitou roli. Je zde vždy snaha o použití co nejmenší velikosti souborů, jelikož čas stažení je přímo úměrný velikosti souborů. U souborů jako je JavaScriptový zdrojový soubor se provádí tzv. minifikace a komprese. [6]

Minifikace je způsob, jak ze souborů kaskádových stylů (CSS) a JavaScriptových souborů odstranit přebytečné znaky. Mažou se především nové řádky, mezery či

poznámky. Jsou to obvykle znaky či symboly, bez kterých je prohlížeč schopen daný soubor přečíst a nebrzdí ho při čtení. Mezi velmi používané nástroje patří například UglifyJS2, Clean-CSS nebo HTML-Minifier. [6], [51]

Kompresí lze dosáhnout zmenšení souborů, např. pomocí algoritmu Gzip. Jedná se o textové soubory, tím pádem HTML, JS, CSS, XML. Pomocí Gzip se soubory na straně serveru zabalí, prohlížeč je následně stahuje zabalené, poté si je rozbalí a následně přečte. Touto metodou lze poměrně dost zrychlit webové stránky. Komprese Gzip se zapíná pomocí Apache modulu mod_deflate do souboru *htaccess*. Pro následné otestování lze použít online nástroj GIDZipTest. [6]

Jak již bylo zmíněno, načítání může být blokováno externími JavaScriptovými soubory. Prohlížeč čeká s vykreslením stránky, než se soubory stáhnou. Blokované načítání lze obejít umístěním odkazů na dané soubory až do patičky stránky. Do tagů <scritp> se umístí atribut async, který způsobí, že se nečeká až se soubor stáhne a prohlížeč pokračuje s vykreslením. [7]

Optimalizace pro soubory typu CSS není tak jednoduchá, jako u externích souborů JavaScriptu. U CSS totiž nelze externí soubory aplikovat až nakonec. Výsledkem by totiž bylo, že by návštěvník webu nejdříve viděl holý text bez jakékoliv grafiky. Po načtení stylů by stránka změnila skokově vzhled – pozicované prvky s různými odsazeními, text by změnil velikost a zarovnání a také by došlo k zbarvení některých prvků. Tento problém lze vyřešit pomocí různých přístupů organizace CSS. [6]

3.2 Organizace CSS

Organizace CSS je velmi důležitá část, které by měl webový vývojář věnovat patřičný čas. I pomocí organizace CSS kódu lze dosáhnout rychlejšího načtení a vykreslení webové stránky. Metodiky pro organizaci CSS slouží především k zpřehlednění a lepší správě kódu stylů CSS.

Společně s organizací CSS se také hodně mluví o preprocesorech, které jsou dnes využívané. Autoři webu CSS-TRICKS již v roce 2016 vytvořili anketu, ve které z třinácti tisíc dotázaných již přes polovinu používá CSS preprocesory. Mezi nejznámější patří preprocesory jako je LESS, SCSS a SASS. LESS využívá 23 %, SCSS 13 %, SASS 5 % z dotázaných. Využití metodik pro organizaci CSS stylů najdeme především u větších či středních projektů. Lze tedy říci, že v týmu je více než jen jeden člověk, a proto právě je

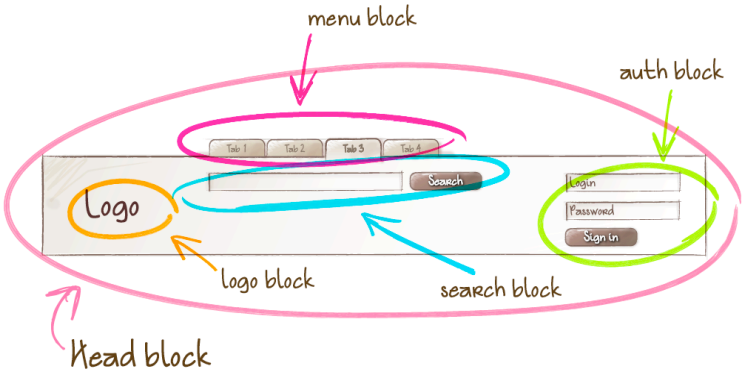
důležité mít nastavená určitá pravidla. Pro menší projekty se nevyplácí tak sofistikovaný a organizovaný kód dělat. Dalo by se říci, že u menších projektů se hlavně sází na organizaci CSS v rámci rychlosti načítání či po logických celcích. [8]

Organizovat CSS lze také podle často se opakujících prvků stránky, podle jednotlivých stránek či jednoduše celý CSS styly u menších projektů nechat v jednom souboru, na který třeba použijeme například kritické CSS pro rychlejší načtení. [9]

3.2.1 BEM – Blok, element, modifikátor

Organizaci lze brát z pohledu zpřehlednění zavedením funkčních pravidel a konvencí při pojmenovávání tříd – například BEM. Lze na něj ale i koukat jako na plnohodnotnou metodiku při organizaci CSS. BEM byl vymyšlen a je dosud používán ve společnosti Yandex. Jeho hlavním přínosem vývojářské komunitě je síla v jeho pojmenovávacích konvencích. Využívá totiž sofistikovaný systém, který velmi usnadňuje jak již zmíněnou znovu použitelnost, tak zavádí pravidla pro velké týmy vývojářů. Systém pojmenování je následující [10], [11], [50]:

- Blok - *.blok*,
- Element - *.blok__element*,
- Modifikátor pro blok - *.blok-modifikátor*,
- Modifikátor pro element - *.blok__element-modifikátor*.



Obrázek 3 - Ilustrace stránky rozdělené do bloků [12]

Blok je nezávislým prvkem, je to část webové stránky tzv. komponenta. Hlavní výhodou je, že se do sebe jednotlivé bloky mohou zanořovat a také jeho znovu použitelnost. Pojmenovávací konvence je zde velmi jednoduchá - *.název-bloku*, kdy víceslovné názvy jsou odděleny pomlčkami. [10], [11], [12], [50]

```
<ul class="nav nav--secondary" role="navigation">
  <li class="nav__item">
    <a href="/">Úvod</a>
  </li>
  <li class="nav__item nav__item--active">
    <a href="/">Produkty</a>
  </li>
  <!-- … -->
</ul>
```

Obrázek 4 - Příklad navigace na stránce pomocí metodiky BEM [11]

Element je prvkem bloku tvoří hlavní a tudíž stěžejní část. Bez elementu by blok nedával smysl a naopak. Je totiž důležité zde dodržet pravidlo existence v rámci bloku. Konvence pro pojmenovávání elementů je pro víceslovné názvy stejná, slova se oddělují čárkami. Ovšem pro návaznost elementu na blok se používají dvě podtržítka - *.název-bloku__více-slovný-název-elementu*. Na výše uvedeném příkladu lze tento element vidět jako položku <li> HTML seznamu. [11], [12]

Modifikátor, jak již název sám napovídá, upravuje konkrétní element či blok. Může upravovat velikost, ale název sám by neměl být závislý na vzhledu – například určovat modrou barvu názvem. Upravuje vizuální vlastnosti, popisuje stavy a chování prvků. Pokud blok či element má mít animaci odjetí nahoru, může vypadat pojmenování modifikátoru takto - *.název-bloku--název-modifikátoru*. [11], [12]

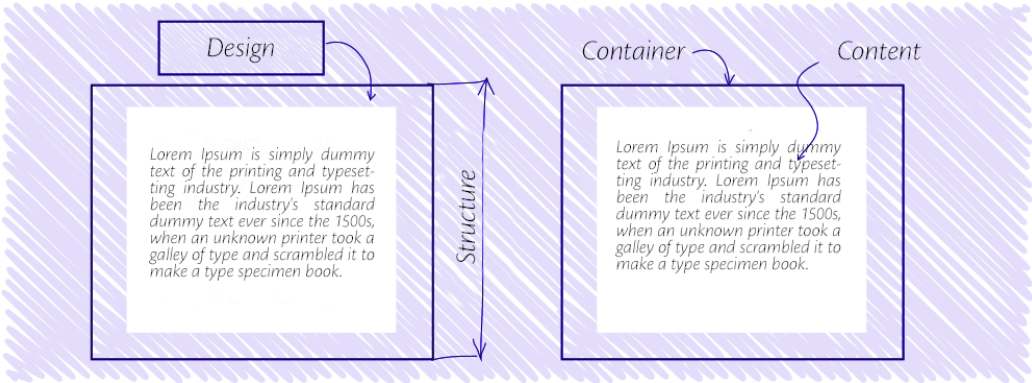
Podle BEM pravidel by se měl název modifikátoru oddělovat pouze jedním podtržítkem. Tento princip takto funguje z důvodu rozlišení dvou typů modifikátorů – hodnotu boolean a klíčovou hodnotu (tj. z angl. key-value). Komunita si ovšem oblíbila verzi Henryho Robertse, kdy zjednodušil modifikátor pouze na jednu variantu – dvou odrážkové verze. [11], [12], [13]

Jak již bylo napsáno, modifikátor lze použít také u elementů. Tato varianta se používá hlavně u obecnějších prvků – komponent. Pravidlo pro pojmenovávání je pak následovná - *.název-bloku__ název-elementu--název-modifikátoru*. BEM metodika je pro některé vývojáře spíše jako doplněk k OOCSS. Je velmi důležitou součástí pouze na úrovni pojmenovávací konvence. [11], [12]

3.2.2 OOCSS

Metodika vznikla v roce 2008 za pomoci webového vývojáře Nicoleho Sullivana. Motivací pro vznik tohoto principu byla inspirace objektovým světem, který známe

například z programovacího jazyku Java či Ruby. Zkratka OOCSS znamená objektově orientované CSS (tj. Object Oriented CSS). Je zaměřená na komponentové psaní a organizaci – tedy nejenom CSS soubory, ale i HTML a JavaScript, který je strukturován do jednotlivých komponent. Často se tato metodika používá samostatně či v kombinaci s SMACSS či BEM metodikou. Jak již bylo zmíněno, výhodou komponentového psaní je znovu použitelnost napříč různými místy v projektu a pokud by se tato praxe dovedla do extrému, lze ji využít také napříč různými projekty. [10], [14]

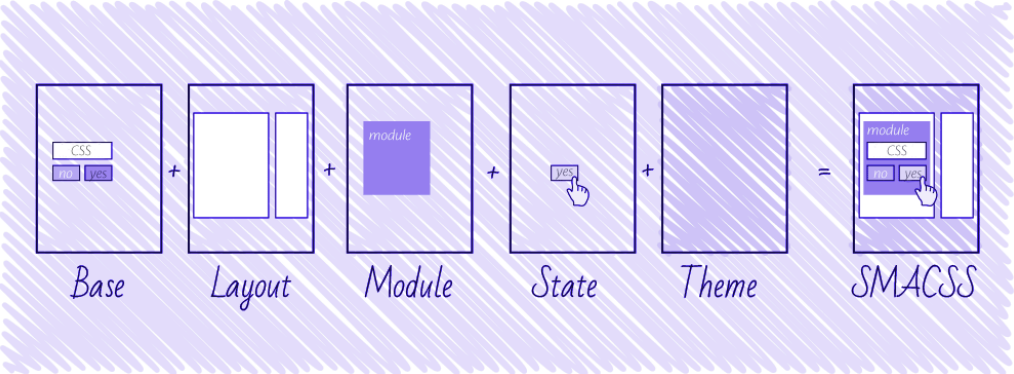


Obrázek 5 - Separace struktury a designu a také obsahu a kontejneru [10]

Mezi další přednosti OOCSS patří také lepší udržitelnost a správa kódu. Jelikož je kód organizován podle jednotlivých komponent, nemůže se stát, že vývojář otevře soubor stylů, ve kterém je styl celé webové stránky. Dalším cílem, který si OOCSS dává, je zmenšení objemu CSS stylů. OOCSS by mělo splňovat pravidla komponentového psaní – tj. nezávislost vzhledu na struktuře, nezávislost obsahu na kontejneru (stejné jako u BEM) a znovu použitelnost. OOCSS lze kombinovat s preprocesory jako je LESS či SASS. [10]

3.2.3 SMACSS

SMACSS je zkratka pro *Scalable and Modular Architecture for CSS*, tedy škálovatelná a modulární architektura pro CSS. Vytvořena byla v roce 2010 Jonathanem Snookem. Hlavním principem této metodologie je rozdělení CSS do pěti hlavních kategorií – Basic, Layout, Module, State a Theme, které jsou znázorněny na schématu níže. [10], [15]



Obrázek 6 - SMACSS se skládá z pěti částí [10]

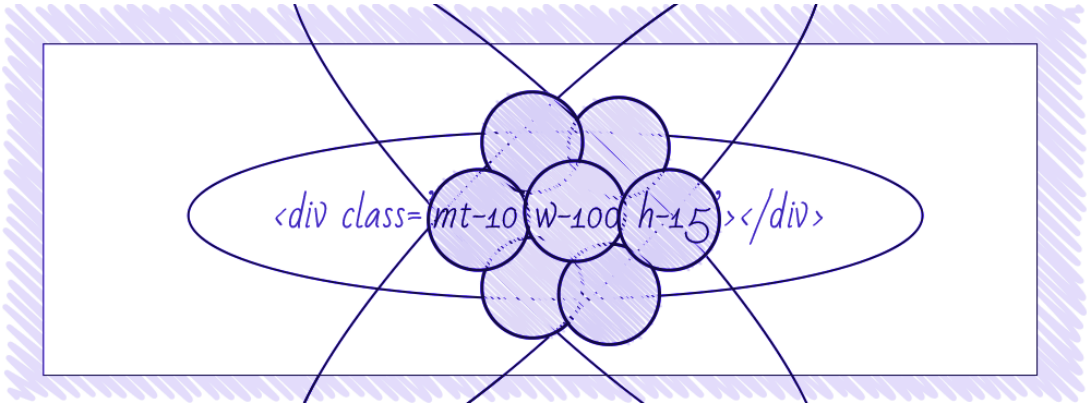
Kategorie Basic (tj. obecná kategorie) obsahuje obecné styly, které lze zapisovat a specifikovat pomocí obecných zápisů jako je např. *input {...}* či blíže specifikovaných atributů, jako je například *input[name="text"]{...}*. Patří sem tedy prvky jako je: body, input, button, ul, ol a další. Také sem patří například resetovací styly jako je Normalize CSS. [10]

Kategorie Layout (tj. kategorie rozložení) obsahuje styly, které rozdělují obsah do určitých sekcí. U této sekce je používána předpona pro označování . *l - název* či . *layout-název*. Kategorie Module (tj. kategorie modul) obsahuje styly, které využívají komponenty a lze je znovu použít. Pro tuto kategorii je doporučeno vyhýbat se selektorům podle id či tagů. U této sekce je používána předpona pro označování . *module - název*. [10]

Kategorie State (tj. kategorie stavů) obsahuje CSS styly, které vypovídají o stavu modulů. Najde se zde např. aktivní stav tlačítka. Je to jediná kategorie, ve které by se mělo používat klíčové slovo *!important*, které slouží k tzv. přebíjení či posílení CSS vlastností. U této sekce je používána předpona pro označování *.is-název*. Kategorie Theme (tj. kategorie vzhledu) obsahuje styly, které je potřeba nahradit za upravené. Tato kategorie je ovšem nepovinná. [10]

3.2.4 Atomic CSS

Systém psaní kódu za pomoci tříd, které mají pouze jediný účel. Tento účel je většinou znát hned z názvu dané třídy. Atomic CSS je také označován jako Utility CSS, ale dá se dohledat i pod jinými názvy. Hlavní princip je ovšem vždy stejný – jednoúčelové třídy. Adam Morse je jedním z autorů Tachyons - tj. jeden z prvních CSS frameworků, který je založen na atomickém přístupu. [10], [16]



Obrázek 7 - Psaní CSS pomocí jednoúčelových tříd [10]

Autor již v roce 2016 psal o datové náročnosti při stahování CSS souborů, upozorňuje na opravdu velkou velikost jednotlivých souborů a s tím spojené ztížení práce. To je podle Adama způsobeno především tím, že soubor obsahuje staré kusy kódu, různé duplicity a další. Právě systém Atomic CSS by tento problém mělo řešit – zamezení duplicitních částí CSS kódu, lepší znovu použitelnost, menší datový objem souborů, lepší přehlednost a údržbu. [16]

Autoři Tachyons proto vytvořili hezký nástroj jménem CSS Stats, který zanalyzuje a vyhodnotí zadanou stránku z pohledu počtu použitých selektorů, tříd a použití různých barev. Zhodnotí také velikost stahovaných CSS souborů a spoustu dalších věcí, které se mohou hodit. [16]

Pokud se to s tím psaním do jednoúčelových tříd přežene, může to dopadnout tak, že vývojář bude psát ke každému prvku 12 tříd. To není úplně ten styl, který utility CSS nebo atomické CSS razí. Kde je tedy rovnováha mezi komponentovým psaním a jednoúčelovými třídami? Martin Michálek ve svém článku o tom, kdy je lepší komponentové psaní a kdy zase použití jednoúčelových tříd, ukazuje několik variant, kdy se za ideální stav označuje psaní komponent pomocí jednoúčelových tříd. Tento systém používá například framework Tailwind CSS. [16], [17]

3.2.5 Komponentní CSS

Komponentní CSS nebo někdy také označováno za progresivní načtení CSS, znamená organizaci stylů a jejich použití na úrovni komponent. Komponentou je zde myšlena velmi malá část stránky – například hlavička, obsah, patička či třeba jen článek. Ve skutečnosti se ovšem stránka neskládá jen z jednotlivých komponent a jejich stylů. V hlavičce stránky jsou tedy jen nejnutnější styly stránky, které se stahují jako první.

Následně dochází postupně k načtení jednotlivých částí – komponent a s tím i stažení jejich stylů. [18]

Jak lze vidět na následujícím kódu, jedná se například o základní strukturu stránky. Soubor layout.css v tagu hlavičky <head> obsahuje základní styly rozmístění prvků stránky jako je hlavička (*header*), obsah (*content*) a patička (*footer*).

```
<html>
  <head>
    <link rel=stylesheet href="layout.css">
  </head>
  <body>
    <link rel=stylesheet href="header.css">
    <div class="header">...</div>

    <link rel=stylesheet href="content.css">
    <div class="content">...</div>

    <link rel=stylesheet href="footer.css">
    <div class="footer">...</div>
  </body>
</html>
```

Obrázek 8 - Příklad komponentního CSS

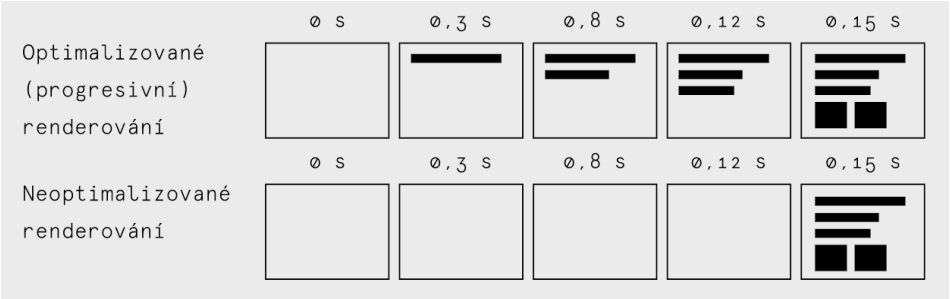
Pro takto organizovanou strukturu CSS stylů je také důležité dodržovat pár pravidel. Pokud je CSS organizován do komponent, měly by styly být nezávislé na struktuře HTML. Komponenta nesmí být závislá na rodiči. Měla by být tzv. znovu použitelná. Co se týká podpory mezi prohlížeči, Internet Explorer a Firefox tento systém podporují již od první verze. I když u prohlížeče se musí použít menší úprava, aby se choval stejně jako IE (Internet Explorer). Od roku 2017 by progresivní načítání stylů měli podporovat i poslední zbývající prohlížeče jako je Chrome společnosti Google a Opera s webovým jádrem Blink a taktéž Safari s jádrem WebKit. [18], [19]

3.2.6 Kritické CSS

„Při prvním vstupu na web totiž návštěvník nepotřebuje ani celé styly pro konkrétní stránku – nepotřebuje například CSS pro patičku webu, když se stejně dívá na horní část stránky. Tento problém řeší takzvané critical CSS.“ [7]

Kritické CSS lze popsat jako rozdělení normálních externích CSS souborů, do několika menších a je zvoleno tak, aby se celky, které jsou vidět jako první, načetly v samostatném dokumentu. Takto je zaručeno, že návštěvník nemusí na začátku stahovat

jeden CSS soubor o velkém objemu dat, ale stáhne si tzv. základní kostru webové stránky či části webu, kterou vidí jako první. Následující obsah je stažen až později. [7]



Obrázek 9 - Styly pomáhají vyplnit čas načtení mezi bílou obrazovkou a načtenou stránkou [20]

U kritického CSS se zaměřuje hlavně na styly, které by mohly zpomalovat načtení webové stránky, proto zde mohou chybět různé odrážky, obrázky či patička kterou uživatel při načtení nevidí. Při načítání stylů asynchronním způsobem může dojít k efektu, kdy návštěvník vidí pouze bílou obrazovku, proto se používá kritické CSS, které spolehlivě zaručí, že se návštěvníkovi nebude zdát po načtení i zbylých stylů tak velký skok mezi jednotlivými procesy a průběh načítání jako celku je plynulejší a pro uživatele příjemnější. [7]

Kritické CSS se generuje různými nástroji, jako je například Critical Path CSS Generator. V dnešní době je ovšem lepší tento proces udělat plně automatizovaným, a proto existují nástroje, které tuto funkci řeší - například criticalCSS. Důležité je zmínit, že načítání kritického CSS by mělo být pouze pro nově příchozí návštěvníky webové stránky. Ti, co se na konkrétní web vrací, funguje velmi dobře mezi paměť v internetových prohlížečích – proto není potřeba stahovat styly znova a tím pádem použít i kritické CSS. Zkoumání rychlosti načtení webových stránek je dnes velmi důležitou praktikou, která může ovlivnit celkovou úspěšnost webu. Stačí pouze zvýšit rychlost načtení o pár milisekund a vlastníkovi internetových stránek to může přinést více klientely a s tím spojené vyšší výdělky. [7], [21]

3.2.7 Jeden CSS soubor

Organizace kaskádových stylů v souborech je jeden z problémů, které řeší weboví vývojáři, frontendisté nebo kodéři. Již od prvních HTML stránek platilo, že CSS pravidla se vkládají do jednotlivých stránek či se připojují do stránky v jednom externím souboru.

V raných dobách webu nedocházelo k přílišnému zkoumání rychlosti načtení a tím pádem ani následné organizaci a optimalizaci CSS. Proto celé webové stránky měly v hlavičce pouze jeden nalinkovaný soubor, který byl minifikován. Tento přístup měl za následek, že první načtení stránky bylo zdlouhavé, ale u dalších stránek došlo k rychlejšímu načtení. Postupem času vznikla potřeba webové stránky zrychlovat. Proto se začala CSS pravidla určitým způsobem organizovat do menších celků či podle určitých pravidel. [22]

3.2.8 Více CSS souborů

Tento způsob organizace kaskádových stylů má velký smysl, pokud vývojář chce na webové stránce nasadit nějaký úhledný systém organizace. Tato organizace CSS pravidel ovšem spočívá ve větší přehlednosti, vytvoření určité struktury (tj. část připomínající komponentové CSS). Rozdělení do menších celků, které spolu logicky souvisejí. Nejčastější případ tohoto užití je, že se označí kritická CSS pravidla (tj. používáno u kritického CSS) a ty se uloží do samostatného souboru, který se načítá první. Dále se oddělí od zbylého CSS také velmi obecná pravidla. Jako další část se může oddělit stylování textů a tak dále. V této organizaci záleží spíše na potřebách daného projektu a také na webovém vývojáři samotném. Při rozdělení CSS pravidel do více souborů dojde k většímu počtu dotazů na stažení daných souborů, tento problém, který by mohl zpomalit načtení lze vyřešit pomocí HTTP/2. [22]

```
<!-- Homepage: -->
<link href="base.css" rel="stylesheet">
<link href="page-home.css" rel="stylesheet">

<!-- Nákupní košík: -->
<link href="base.css" rel="stylesheet">
<link href="page-cart.css" rel="stylesheet">
```

Obrázek 10 - Více CSS souborů v hlavičce webové stránky [22]

3.2.9 Shrnutí

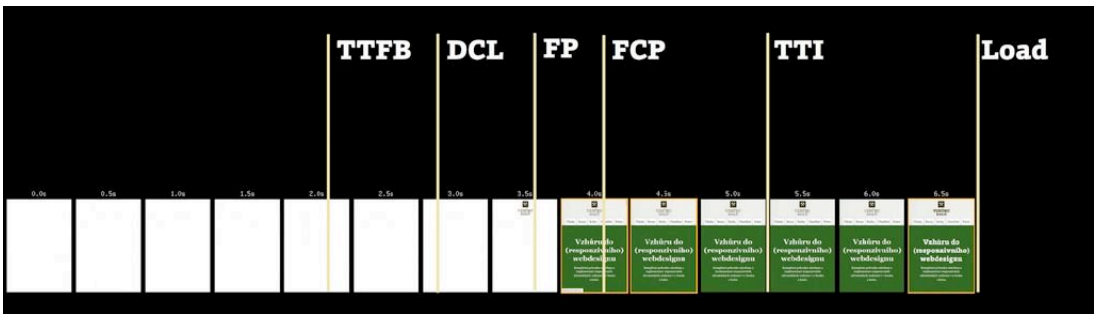
Je nutno podotknout, že pokud se rozhodne vývojář pro jakoukoliv metodiku organizace CSS, mělo by být vždy myšleno především na jednoduchost a nesnažit se za každou cenu nasadit jednu z daných metodik. Ne vždy je totiž tento postup správný.

Dokonce může dojít k scénáři, kdy například vývojář použije BEM metodiku pro pojmenování a OOCSS pro organizaci.

Po představení spíše komplexních metodik a pravidel pro psaní a organizaci CSS, došlo k ukázce také jiného pohled na věc. Všechny předchozí metodiky až na jednoúčelové třídy se totiž hodí spíše pro větší projekty – pro agilní vývoj a samozřejmě pro objektový svět. Systémy organizace jsou další z možností, jak lze ve stránce organizovat CSS. Ovšem jsou více obecné než dosud zmíněné a řeší spíše problém blokování vykreslování obsahu stránky.

3.3 Měření rychlosti načítání

Měření rychlosti načtení webových stránek lze provádět pomocí různých metrik. Jak lze vidět na obrázku č. 10, jde o jednotlivé milníky, které jsou při načítání webové stránky pozorovány. Měření a zkoumání rychlosti vykreslení stránky se zabývá také marketing jakožto obor, který se snaží nejenom o prezentaci, ale také i o co nejlepší uživatelský dojem (UX tj. user experience) na webových stránkách. [23], [24]



Obrázek 11 - Jednotlivé metriky seřazeny podle načítání webové stránky [24]

Uživatele můžeme odradit dlouhým načítání a nemusí tak dojít ke chtěné konverzi. Pokud má návštěvník interakci s webem podle doby načtení [4]:

- menší jak 100 milisekund – uživatel nečeká a považuje to za okamžitou reakci,
- menší jak 1 sekunda – uživatel pozná prodlevu, ale nepřeruší úkol,
- více než 10 sekund – uživatel ztrácí pozornost a přesouvá se k jiné činnosti.

Proč je dobré řešit rychlost načtení? Řetězec obchodů Wallmart zjistil, že každé zrychlení o jednu sekundu pro něj znamená zvednutí konverzí (tj. dokončení cíle návštěvníka na webu) návštěvníků o 2 %. Také vyhledávače jako je Google dnes preferují ve vyhledávacích výsledcích stránky, které mají rychlejší načítání. [23], [24]

Rychlost načtení je důležité řešit také z pohledu používání mobilních zařízení, v tomto roce je to již 10,8 % z celkových návštěv. Meziroční nárůst oproti roku 2018 je o 1,3 %. Jak je již známo, internetové připojení skrze vzduch se nikdy nevyrovná tomu pevnému na počítači, proto je potřeba objem stahovaných dat redukovat a optimalizovat pro mobilní zařízení za pomocí měření. [23], [25]

3.3.1 Druhy měření

Nemusí se nutně sledovat pouze rychlost načtení kompletní stránky, ale třeba například prvního snímku, který je pro uživatele smysluplný. Metriky lze získat dvojím způsobem – syntetickým měřením nebo měřením reálných uživatelů. [26]

3.3.1.1 Syntetické měření

Syntetické měření, je takové měření, při kterém robot projde web. Tento robot má na starost simulaci reálného uživatele. Jedná se tedy většinou o plně automatizovanou službu, která také měří rychlost internetu a zkoumá konkrétní prohlížeč.

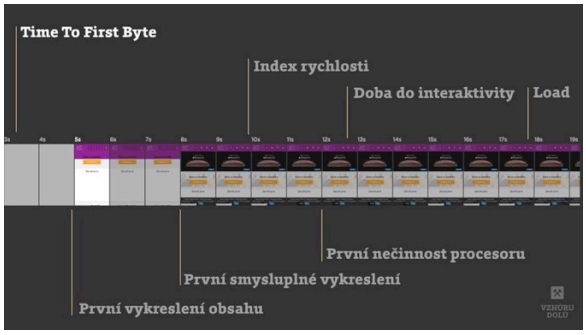
Tato technika analýzy je používána nejčastěji. Syntetické měření využívají hlavně nástroje jako jsou Lighthouse, PageSpeed Insights, WebpageTest.org. Většina z těchto nástrojů je dostupných online a zdarma, ovšem každý z nich může jednotlivé sledované metriky interpretovat pod jinými názvy či vůbec. [26]

3.3.1.2 RUM měření

Měření reálných uživatelů pochází z anglického překladu Real User Monitoring (tj. RUM). Tento typ měření je používaný méně, ovšem je to o dost přesnější metoda získávání dat. Jak je již z názvu patrné, není zde žádný robot, ale k získávání dat jsou využíváni skuteční návštěvníci webových stránek – tedy reální uživatelé. Nejčastěji se tyto nástroje napojují do stránek pomocí vložených skriptů. Tyto nástroje jsou mohou být placené, proto je jejich obliba a používání daleko menší než u výše zmíněné metody. Například služba SpeedCurve, nabízí měření jak RUM (tj. real user monitoring), tak syntetické. Nástroj PageSpeed Insights od Google používá kombinaci syntetického měření v kombinaci s reálnými daty uživatelů. [26]

3.3.2 Metriky

Metrika by se dala definovat jako číselný údaj, který vystihuje určitou událost v čase. Následující metriky jsou seřazené, jak k nim dochází při načtení webové stránky.



Obrázek 12 - Jednotlivé metriky přehledně zobrazené při načítání webové stránky [24]

Metriky slouží pro optimalizaci rychlosti webové stránky nebo také k optimalizaci stránek z pohledu uživatelského zážitku (tj. UX – user experience).

3.3.2.1 Metrika Time To First Byte

Time To First Byte neboli zkráceně TTFB v překladu znamená – „doba do načtení prvního bytu“. Tato metrika nám udává, jak rychle stáhne prohlížeč první byte HTML stránky. Tato metrika se používá především pro určení rychlosti backendové části a také rychlosti serveru. Nejčastěji je označována jako metrika rychlosti serveru. Také vypovídá o rychlosti sítě, mezi serverem a klientem. Jak zmiňuje Martin Michálek ve svém článku, u této metriky často vzniká mýtus, že rychlost načtení webu zásadně ovlivňuje rychlost serveru. Tato metrika není až tak důležitá, ovšem neměla by přesáhnout hodnotu půl vteřiny. [27]

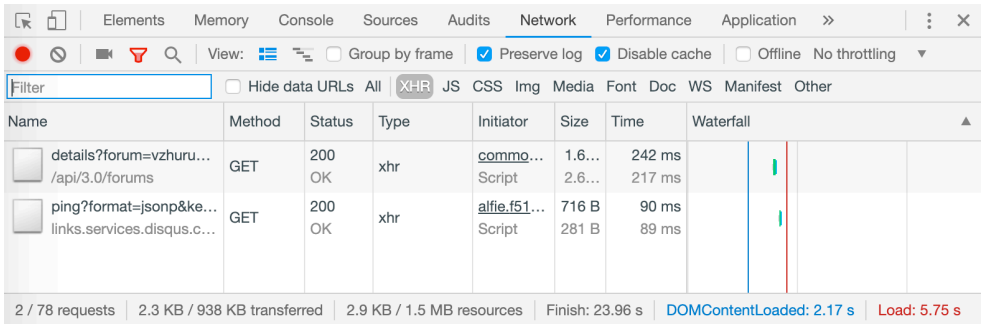
Tato metrika nepřímo souvisí také s termínem *Crawl budget,* neboli označením pro počet stránek, které vyhledávače, tedy jejich robotický crawleři navštíví během procházení webu. Tento termín je velmi hojně používán v marketingu při práci se SEO prvky – tj. search engine optimalization, neboli optimalizací webů pro vyhledávače. Týká se to hlavně větších webů, kdy je třeba, aby crawler prošel co největší počet stránek za čas, který má přidělený na prozkoumání určitého webu. [27]

TTFB lze taktéž měřit pomocí mnoha rozdílných nástrojů jako je PageSpeed Insights, WebpageTest.org nebo placený nástroj SpeedCurve. Při provádění měření musí být bráno v potaz, že aspektem zkoumání se může stát pomalejší internetové připojení

nebo náhlá vytíženost serverů. Proto je spíše doporučeno sledovat u této metriky dlouhodobý vývoj než se snažit dosáhnout, co nejnižší hodnoty. [27]

3.3.2.2 Metrika DOM Content Loaded

DCL neboli DOM Content Loaded je typ měření, které je vázáno událostí DOMContentLoaded z JavaScriptu. Tato událost je vytvořena, když je hlavní HTML dokument stažený a rozparsovaný, tedy nečeká na žádné další potřebné prvky. Ve vývojářských nástrojích lze tuto metriku vidět jako modrou čáru v záložce Network. [28]



Obrázek 13 - V záložce Síť (Chrome) je tato metrika reprezentována modrou čárou

Metrika DCL není v obrázku č. 12 zobrazena, ovšem nachází mezi „Prvním vykreslením obsahu“ a „Prvním smysluplným vykreslením“. [28]

3.3.2.3 Metrika First Paint

FP neboli First Paint je metrika, která je definována, jako okamžik, kdy je od kliknutí uživatele prohlížečem vyrendrováná jen část, která je odlišná od předchozí obrazovky. Tato metrika není příliš používána, daleko více se preferuje následující typ, jelikož vypovídá o tom, že se začal vykreslovat obsah, nikoliv jen obecné vykreslení. [29]

3.3.2.4 Metrika First Contentful Paint

FCP neboli první vykreslení obsahu. Je metrika, která vypovídá o prvním vykreslení obrázku či textu. Tato metrika je důležitá z pohledu uživatele, říká mu totiž, že se opravdu něco začíná načítat. Tuto metriku lze pozorovat v nástrojích, jako je již zmíněný Lighthouse a také v PageSpeed Insights. [30]

Pro zlepšení této metriky, tedy snížení její hodnoty, přímo společnost Google ve svém programu Lighthouse doporučuje zbavení se aspektů, které blokují vykreslení DOM prvků. Je doporučeno minimalizovat a zmenšit velikost stahovaných souborů, zapnout

zapisování souborů do mezipaměti a načítat JavaScriptové externí soubory třetích stran až na konci stránky. [30], [31]

3.3.2.5 Metrika First Input Delay

FID neboli metrika první nečinnosti procesoru. Tato metrika nám uvádí, jak dlouho uplyne od startu interakce uživatele a jejím skutečným provedením. Tuto metriku velmi ovlivňuje JavaScript. FID je velmi blízko metrice TTI (tj. pokud seřadíme metriky, jak jdou v čase), ovšem oproti TTI metrice vypovídá o reálné uživatelské zkušenosti. Provádí se za pomocí měření RUM, v nástrojích jako jsou Chrome UX Report nebo SpeedCurve. [32]

3.3.2.6 Metrika Meaningful Paint

FMP neboli metrika prvního smysluplného vykreslení. Vypovídá o prvním vykreslení stránky, při kterém jsou vidět hlavní obsahové části stránky. Většinou se jedná o vykreslení hlavního prvku na dané stránce – nadpisu, video, obrázku. Tuto metriku ukazují nástroje Lighthouse a PageSpeed Insights. U této metriky by mělo být zmíněno, že dosažením co nejmenší hodnoty lze docílit optimalizováním kroků které souvisí s tzv. kritickou vykreslovací cestou. [33]

3.3.2.7 Metrika Time to Interactive

TTI neboli čas do interaktivity je metrika, která říká, kdy je stránka připravena na uživatelský vstup. Tato situace většinou nastane po načtení a spuštění JavaScriptů, které jsou na webové stránce používány. Tuto metriku lze změřit například v nástroji Lighthouse, kde je označována jako Consistently Interactive a v nástroji WebpageTest.org označována jako First Interactive. [34]

3.3.2.8 Metrika Speed Index

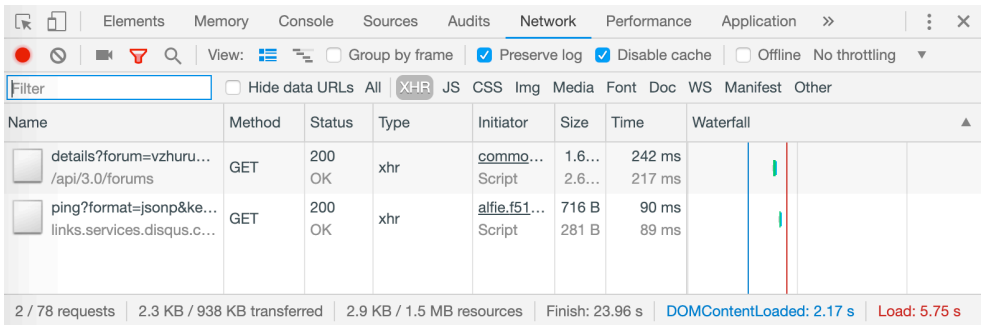
Přeloženo jako index rychlosti. Tato metrika znázorňuje hodnotu, jak dlouho zabere vykreslení do úplného zobrazení. Tato metrika je hodně ovlivňována použitými technologiemi – prohlížečem, rychlostí připojení anebo také velikostí okna. Tato metrika neukazuje čas, ale hodnocení webu. Čím nižší hodnota, tím lépe pro zkoumanou webovou stránku. Tuto metriku ukazuje především nástroj WebpageTest.org. V nástroji Lighthouse tuto metriku lze nalézt pod názvem Perceptual Speed Index. [35], [36]

Celý princip této metriky je postavený na základě natočeného záznamu načtení webové stránky. Lze tak usoudit, že tato metrika nebude mít ideální hodnotu, pokud vývojář použije na načítání nějaký efekt, jelikož nástroj čeká na kompletní vykreslení. Tato metrika lze zlepšit nejčastěji vykreslením pouze kritických zdrojů, nekritické zdroje se odloží na pozdější vykreslení. Dále je velmi účinné zbavit se všeho nepoužitého. [35], [36]

Za ideální hodnotu lze považovat hodnotu okolo jednoho tisíce. Jako průměr se považuje hodnota od pěti do deseti tisíc. Hodnoty vyšší, jak deset tisíc lze považovat za alarmující. Tato metrika se používá také pro srovnávání s konkurencí či pro srovnání stavu webu před a po optimalizaci. [35], [36]

3.3.2.9 Metrika Load

Tato metrika vypovídá o načtení veškerého obsahu stránky. Lze ji nalézt ve vývojářských nástrojích v prohlížeči Chrome i Firefox v záložce Network. Je označená červenou čárou. [37]



Obrázek 14 - Metrika Load je znázorněna červenou čárou v záložce Síť

3.3.3 Nástroje pro analýzu rychlosti

Pomocí těchto nástrojů lze provádět analýzu rychlosti načtení webu a následnou optimalizaci. Na jakém principu jsou založené tyto nástroje se čtenář dozví v následujících odstavcích. Při testování by uživatel měl otestovat všechny vstupní stránky webu. To znamená nejenom hlavní stránku, ale všechny stránky, na které se návštěvníci dostávají na náš web.

3.3.3.1 PageSpeed Insights

PageSpeed Insights neboli PSI je online nástroj od společnosti Google. Jedná se o jeden z nejznámějších a nejpoužívanějších nástrojů mezi vývojáři. Tento nástroj kombinuje

jak syntetické meření, tak měření reálných uživatelů – alespoň tedy data od reálných uživatelů. [26], [38], [39]

Nástroj nejdříve provede syntetické měření, v kterém simuluje pomocí počítače návštěvníka webu. Po dokončení testu nám nástroj vyhodí skóre, které je sestaveno na základě tří kategorií [26], [38], [39]:

- od 100 do 90 bodů je stránka považována za rychlou,
- od 90 do 50 bodů je stránka považována za průměrně rychlou,
- od 50 níže bodů je stránka považována za pomalou.

Po provedení tohoto měření se podívá do databáze Chrome UX Report, což je databáze údajů, které jsou změřený reálnými uživateli prohlížeče Chrome. Zde nástroj měří FCP neboli již zmíněnou metriku první vykreslení obsahu, také FID neboli první nečinnost procesoru. Tyto dvě metriky Google také kategorizuje a to následovně [26], [38], [39]:

	Fast	Average	Slow
FCP	[0, 1000ms]	(1000ms, 2500ms]	over 2500ms
FID	[0, 50ms]	(50ms, 250ms]	over 250ms

Obrázek 15 - Kategorizace rychlosti načtení webu podle dvou metrik FCP a FID [26]

Google u těchto hodnot uvádí, že jím označované rychlé (tj. Fast) webové stránky je pouze 10 % ze všech měřených. Do průměru (tj. Average) spadá dalších zhruba 40 % a zbylých 50 % měřených stránek je prostě pomalých. Na základě provedeného měření nám nástroj vytvoří výsledky měření pro mobilní zařízení a také pro klasický počítač. Výsledky se skládají ze změřených metrik jako jsou: První vykreslení obsahu, Index rychlosti, doba do interaktivity, první smysluplné vykreslení, první nečinnost procesoru a maximální potenciální prodleva prvního vstupu. Dále je zde také zobrazený přehled snímku vykreslení stránky v čase – lze použít při řešení kritického CSS. [26], [38], [39]

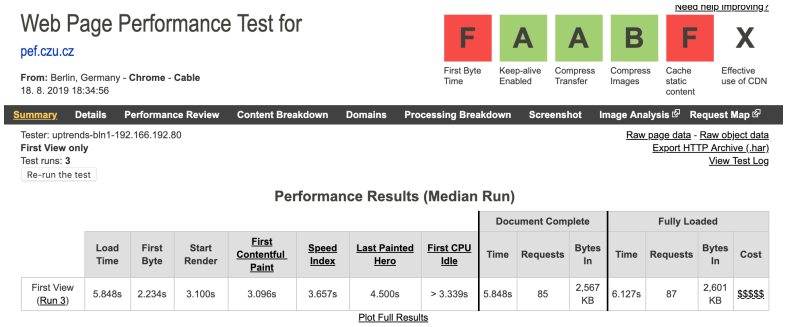
Obrázek 16 - Přehled snímku vykreslení stránky Provozně ekonomické fakulty ČZU [40]

V záložce Příležitosti jsou body, které mohou na základě navržených optimalizací, zrychlit web. V záložce nazývané Diagnostika lze nalézt další seznam bodů, který vypovídá o výkonu webu. [26], [38], [39]

Nástroje jako je PSI od Google musí každý vývojář, který se zabývá rychlostí načtení webu, brát trochu s rezervou. Na ideální skóre 100 bodů ze 100 lze dosáhnout jen velmi těžko nebo vůbec. Je to způsobené vyhodnocením špatně nastaveného kešování externí skriptů třetích stran – ovšem s tímto nic udělat nelze. Proto by vývojáři měl tento nástroj sloužit k odhalení těch největších chyb nikoliv všech. Dá se říci, že jej lze použít jako výchozího bodu pro začátek analýzy a na základě změřených výsledků se za pomoci detailnějších měření dobrat k finálním hodnotám. [41]

3.3.3.2 Webpagetest.org

Online nástroj Webpagetest.org dává mnohem větší objem naměřených informací. Oproti PageSpeed Insights, nástroj informace poskytuje pro vývojáře v méně přehledné formě. Již na začátku musí vývojář nastavit lokaci z které probíhá měření a také zařízení. Webpagetest.org poskytuje základní informace o metrikách. Tím je tabulka s časy a také hodnocení jednotlivých metrik písmenky od A až po F. [39], [42]



Obrázek 17 - Puštěný test webu Provozně ekonomické fakulty ČZU [43]

Výsledná analýza obsahuje vodopádový graf, který zobrazuje, zda nějaký ze souborů neblokuje načítání stránky. Nástroj také poskytuje pás snímků při vykreslování stránky v čase, což je velmi užitečné při optimalizaci vykreslení pro nově příchozí návštěvníky webu, kde se využívá kritického CSS. [39], [42]

Obrázek 18 - Pás snímků, který zobrazuje proces vykreslení do úplného vykreslení stránky [43]

Testy probíhají celkem tři. Ze všech tří testů je následně složena výsledná tabulka, která obsahuje zprůměrované hodnoty. V tomto nástroji najdeme daleko více informací,

ovšem PageSpeed Insight umí analyzované informace daleko lépe interpretovat směrem k vývojáři. Webpagetest.org ukazuje mnoho dalších hodnot, jako je Speed Index, počet DOM prvků, počet dotazů, celkový čas načtení webu a mnoho dalších. [39], [42]

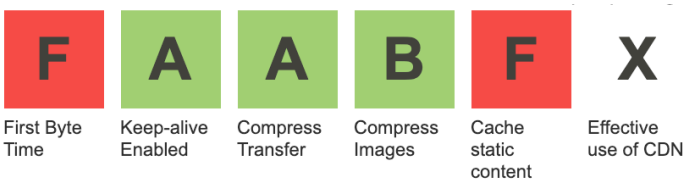
Performance Results (Median Run)

	Load Time	First Byte	Start Render	First Contentful Paint	Speed Index	Last Painted Hero	First CPU Idle	Document Complete			Fully Loaded			
								Time	Requests	Bytes In	Time	Requests	Bytes In	Cost
First View (Run 3)	5.848s	2.234s	3.100s	3.096s	3.657s	4.500s	> 3.339s	5.848s	85	2,567 KB	6.127s	87	2,601 KB	$$$$$

Plot Full Results

Obrázek 19 - Tabulka zprůměrovaných hodnot tří po sobě jdoucích testů [43]

Nástroj také umožňuje testování webu z jiné lokality, což často vede k odhalení pomalého načtení z různých lokací po celém světě. Tento problém lze vyřešit využitím tzv. CDN pro celý web. Lze vidět na obrázku č. 16, že nástroj detekuje, zda web využívá efektivitu CDN. [39], [42]

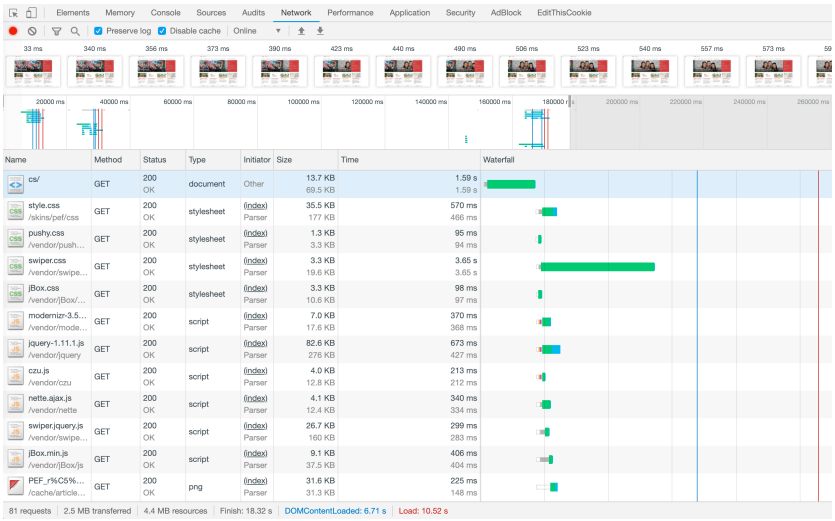


Obrázek 20 - Výsledné hodnocení vybraných ukazatelů a jejich hodnocení na škále od A po F [43]

Dále lze také provést simulování pomalého připojení k internetu či si vybrat z různých internetových prohlížečů. Nástroj se snaží o výsledné skóre, kdy autoři tohoto nástroje vybrali parametry jako je „First Byte Time“ (tj. rychlost serveru). Skóre je na škále od A (tj. nejlepší) po písmeno F (tj. nejhorší). [39], [42]

3.3.3.3 Chrome DevTools

Tento nástroj je integrován do prohlížeče Google Chrome. Do nástroje se dostaneme pomocí ikony hlavního menu vpravo nahoře, zvolíme *Další nástroje*, poté *Nástroje pro vývojáře*. Veškeré informace potřebné k analýze rychlosti načtení stránky se nacházejí v záložce Network (tj. Síť). [39]

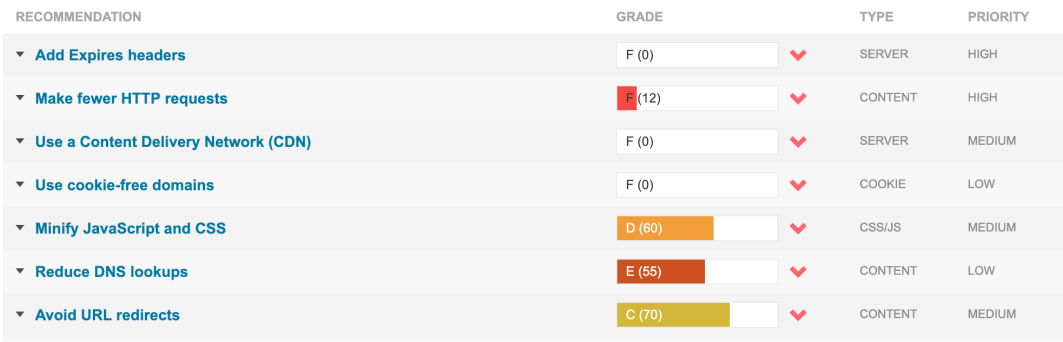


Obrázek 21 - Nástroj pro analýzu rychlosti načtení se nachází v záložce Síť (z angl. Network)

Před načtením stránky je důležité zapnout nahrávání pro viditelný pás snímků. Dále může uživatel omezit rychlost načítání na rychlosti jako je klasické, rychlé 3G či pomalé 3G. Vlastní profil rychlosti načítání lze vytvořit velmi jednoduše.

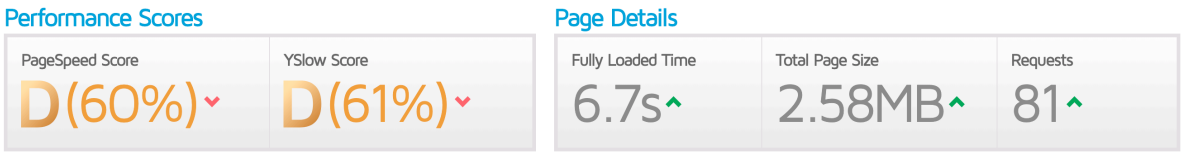
3.3.3.4 GTmetrix

Nástroj provádí syntetické měření, jelikož simuluje uživatele tak, jak již bylo zmíněno výše v kapitole zaměřené na typy měření. GTmetrix využívá metodiku YSlow. Tato metodika vznikla za pomoci vývojářů Yahoo!, kteří sepsali třicet čtyři pravidel, které ovlivňují výkon webu – tedy i rychlost načtení. Výsledné skóre je uděleno na základě dvaceti tří pravidel, těch nejdůležitějších a pro web nejpodstatnějších, z celkového počtu třiceti čtyř. Mezi hodnocená pravidla patří: minimalizování HTTP požadavků, používání CDN, používání GZipu, minifikování skriptů a stylů, vyhnutí se určitým výrazům v CSS, změnšení počtu prvků DOM stromu a mnoho dalších. [39]



Obrázek 22 - Výsledky webu Provozně ekonomické fakulty ČZU podle metodiky YSlow [44]

Služba GTmetrix vyžaduje registraci pro zpřístupnění všech měřených metrik a dalších funkcionalit. Nástroj umí testovat různé lokality, ovšem pro ČR jsou oproti Webpagetest.org horší. Je zde emulátor pomalého připojení, ukáže nám časový pás snímků načtení webové stránky a umí zobrazit vodopádový graf. [39]



Obrázek 23 - Ukázka výsledného skóre pro web Provozně ekonomické fakulty ČZU [44]

Nástroj ukazuje skóre výkonu pomocí metodiky YSlow a Pagespeed. Také ukazuje celkové statistiky ohledně objemu stažených souborů, počtu dotazů či metriky celkové načtení. Obě dvě metodiky mají známkování jednotlivých bodů na stupnici od A po F (tj. nejhorší). [39]

3.4 Vykreslovací jádro

Vykreslovací jádro je nástroj, který vykreslí text a obrázky na obrazovku návštěvníka webových stránek. Nástroj bere obsah, který kreslí nejčastěji ze strukturovaného dokumentu formátu HTML, následně dochází k stylování jednotlivých strukturovaných částí, za pomocí deklarovaných stylů. Deklarované styly jsou nejčastěji uvedeny ve formátu CSS. Následným úkolem po vykreslení stránky je překreslovaní při různých akcích návštěvníka webových stránek. Pro jednodušší pochopení akce, je myšleno například návštěvníkovo najetí myší na odkaz, který se zbarví do modra a podtrhne. Kliknutí na tlačítko, které vyvolá například JavaScriptovou akci, která následně do stránky vykreslí další obsah. [45]

Vykreslovací jádro je jedna z podstatných částí webového prohlížeče. Má ho každý internetový prohlížeč. Bez jádra by prohlížeč nebyl schopen interpretovat zdrojové kódy, které mu po zadání webové adresy stránky (URL) vrátí webový server. [45]

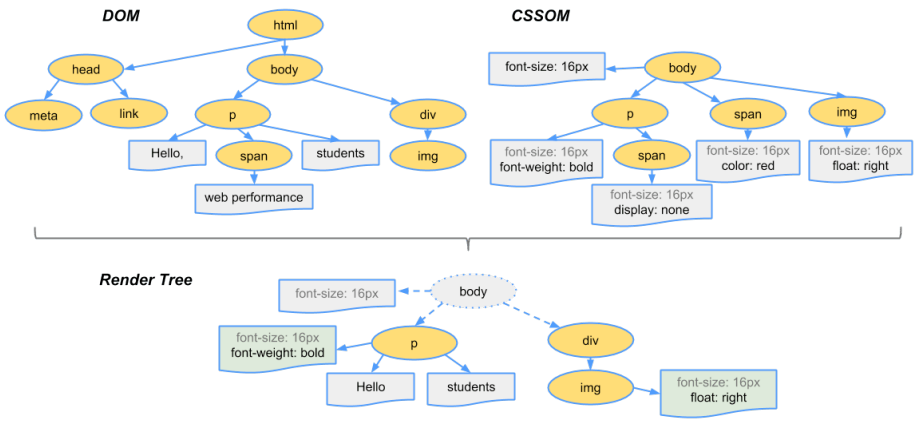
Většina z nich má ovšem tato jádra rozdílná, tedy princip sestavení stránky ze zdrojových souborů se může lišit podle použitého jádra. Mezi nejčastěji používané prohlížeče patří Chrome, Safari, Internet Explorer & Edge, Firefox a Opera. Prohlížeč Chrome a Opera používají stejný vykreslovací jádro, které se nazývá Blink. Prohlížeč Safari od společnosti Apple používá zase vykreslovací jádro nazývané WebKit. Jádro

Blink vychází z WebKitu, tím pádem mají některé části vykreslovacího procesu stejné. Prohlížeč Internet Explorer používá Trident, Edge používá EdgeHTML a jako poslední Firefox využívá Gecko. [45], [47]

3.4.1 Proces vykreslování

Jak již bylo zmíněno, celý proces začíná u zadaní adresy webové stránky do prohlížeče. Poté se stáhnou data potřebná k vykreslení. Tato data jsou HTML kód a styly.

Nejdříve je sestaven tzv. DOM neboli Document Object Model – strom HTML prvků. Tento strom je často označován, jako zdrojový kód stránky. Tomu tak ovšem není, DOM totiž obsahuje některé značky, které ve zdrojovém kódu nejsou zapsány. Pokud bychom řešili principiálně tento problém, je to spíše naopak, kdy ze zdrojového kódu je vytvořen DOM, kdy se vezmou všechny HTML prvky stránky a sestaví se do DOM stromu. [45], [46]



Obrázek 24 - Podoba DOM, CSSOM a výsledného vykreslovaného [48]

Například v prohlížeči Google Chrome jej můžeme najít v záložce Vývojář, hned jako první kartu. Při sestavování DOM stromu se doplňují nepovinné značky koncové či počáteční jako jsou [46]:

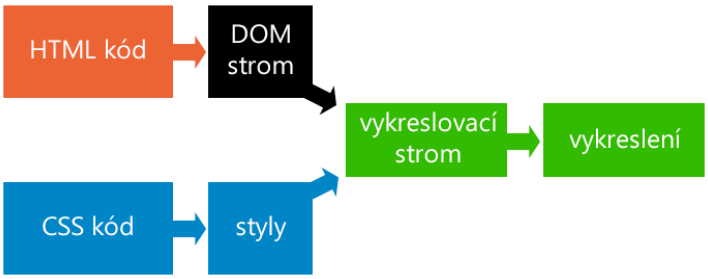
- značky <html>, <head>, <body>, <tbody>,
- značky odstavců <p>,
- značky seznamů <li>, <dd>, <dt>,
- značky řádků a buněk tabulky <tr>, <td>, <th>,
- značka <option> pro prvky formuláře.

Dále se také opraví vzniklé chyby v DOM stromu, jako jsou například překřížené značky. V druhém kroku se udělá obdobný proces jako při sestavování DOM stromu,

ovšem se sestavuje z CSS zase CSSOM neboli CSS Object Model – tento model je tvořen smícháním následujících stylů [46]:

- styly prohlížeče,
- externí a interní styly CSS,
- inline styly a atributy HTML prvků,
- v poslední řadě uživatelské styly.

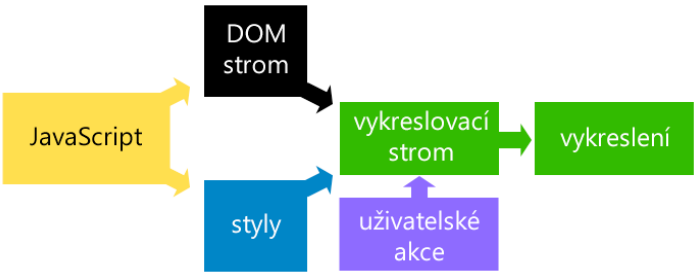
V třetím kroku se z HTML stromu DOM a CSS vytvoří vykreslovací neboli rendrovací strom, v kterém jsou již aplikovány CSS styly na všechny jeho prvky. Například, kdyby nějaký prvek odstavce v zápisu stylu měl uvedenou vlastnost *display: none*, dojde k tomu, že se tento prvek nedostane vůbec do výsledného renderovacího stromu. Po předchozím sestavení vykreslovacího stromu tedy nebrání již nic k výslednému vykreslení webové stránky prohlížečem. [45], [46]



Obrázek 25 - Pro rekapitulaci postupu vykreslení stránky pomocí jádra [46]

3.4.2 Překreslovací proces

Tento proces je určen hlavně ke změnám stránky. Změny se většinou provádějí pomocí JavaScriptu. Webová stránka většinou reaguje na uživatelské akce, které jsou prováděny uživatelem. Napsaný JavaScriptový kód pak zaručí, že se změní DOM strom či přepíše stylování jednotlivých HTML elementů. [46]



Obrázek 26 - Postup překreslení [46]

Rozlišují se dva druhy překreslení – *Layout* a *Paint*. *Layout* neboli *reflow*, je typ překreslení, kdy dochází znovu k sestavení části nebo celého vykreslovacího stromu. Tato situace se stane obvykle pokud prvek ovlivní své okolí. Dochází k tomu také při [49]:

- přidání, odstranění či změna elementů do DOM stromu,
- přidání, odstranění či změna CSS stylů,
- CSS3 a jeho animace (z ang. *animation*) a přechody (z ang. *transition*)
- při skrolování a změně velikosti okna.

Paint neboli *redraw* je typ překreslení, při kterém nutně nemusí dojít k znovu sestavení vykreslovacího stromu. Například tomu tak je při změně barvy pozadí prvku nebo změně průhlednosti.

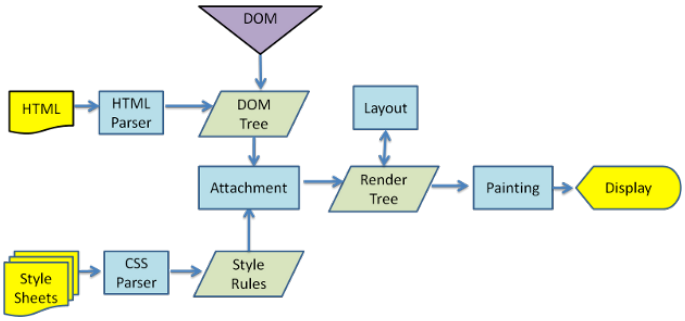
U CSS vlastností dochází k typu překreslení v závislosti převážně na prohlížeči a jeho vykreslovacím jádru. Jak již bylo zmíněno výše, každé vykreslovací jádro se chová trochu jinak. Proto v návaznosti na to, se pak určitý typ překreslování spouští u vlastností jinak. Tento jev je označován jako CSS spouštěč (z anglického CSS Triggers). [45]

Jelikož by tento proces stál hodně výkonu, prohlížeč by si vzal výpočetní prostor. Vývojáři se snaží optimalizovat proces převážně při překreslování, kdy se při změně jednoho prvku hned nevykresluje znovu celá stránka. Změny, které má na starosti JavaScript, se shlukují do fronty, ty se pak provedou naráz a stránka se následně překreslí. Rolování stránky – jedna z častých akcí, které musí prohlížeč zvládnout překreslit. Prohlížeč se tedy snaží odhadnout, které části DOM stromu budou vypadat stejně a které se změní. Rozdělí se do tzv. skupin, které pak postupně buďto překreslí nebo ponechají. Pro optimalizaci a plynulost rolování stránky, se proto v tom nejlepším případě snaží prohlížeč překreslit co nejméně skupin. [45]

Velmi problematické jsou prvky, které jsou velké a zafixované. Jádro prohlížeče totiž musí ještě zjišťovat co ve všech těchto skupinách je překrýváno a co naopak není. Jako další efekt, který je velmi hojně používán je tzv. *parallax*, kdy se při rolování stránky posouvají i některé prvky. Z pohledu vykreslovacího jádra je tento scénář nejhorší, který může nastat. Při posunutí stránky se pokaždé musí složitě zjišťovat posuny, zda nějaký z prvků nepřekrývá jiný a tak dále. [45]

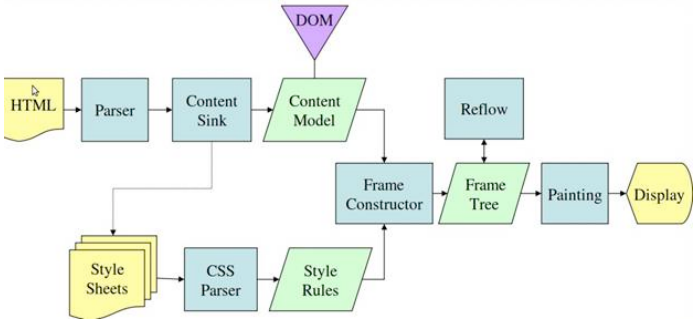
3.4.3 Porovnání vykreslovacích jader

Pro srovnání vykreslovacích jader byli vybrány WebKit a Geco. Jak již bylo zmíněno, WebKit je vykreslovací jádro prohlížeče Safari od společnosti Apple, z tohoto jádra vzešel také Blink. [45]



Obrázek 27 - Proces vykreslení jádra WebKit [45]

Naproti tomu jádro Gecko je vykreslovací jádro prohlížeče Firefox od společnosti Mozilla. Jak lze vidět na obrázku č. 26 a č. 27, obě vykreslovací jádra mají lehce jinou terminologii v celém procesu vykreslení. Při podrobném zkoumání lze vidět, že vykreslovací jádro Gecko má o jeden krok více. Tento krok se nazývá v angl. Content Sink a funguje jako továrna na výrobu prvků DOM stromu. Obecně lze říci, že obě jádra mají proces vykreslení stejný. [45]



Obrázek 28 - Proces vykreslení jádra Gecko [45]

3.4.4 Optimalizace překreslování

Překreslování je velmi podstatný a někdy velmi náročný úkon pro jádro webového prohlížeče. Existují určitá pravidla, která když budou následována, dojde k optimalizaci a zamezení zbytečnému překreslování celé stránky prohlížečem. Obecně lze říci, že by se neměly používat "*inline styly*", což jsou CSS styly, které jsou psány přímo k HTML tagům do atributu *style*. Jako příklad inline stylů lze uvést například [49]:

text

Jako další by se neměl používat tzv. tabulkový layout, pro zobrazení prvků bude použita CSS vlastnost table-layout, která způsobuje, je sice stránka vykreslena, ale při každé buňce tabulky se bude překreslovat, kvůli výpočtu rozměrů. Pokud je nadále potřeba používat tabulkové layouty, lze nastavit vlastnosti table-layout na hodnotu fixed. Při vykreslování se změní chování z překreslování u každé buňky, na překreslení celé tabulky, podle známé šířky samotného záhlaví. [46], [49]

Pro optimalizaci a zrychlení překreslování je doporučeno také minimalizovat počet CSS pravidel, ať už jsou tato pravidla v externím CSS souboru či přímo vložené ve stránce, pomocí tagu <style>. Nejčastěji je toto problém frameworků jako je Bootstrap, Foundation a další. Pro optimalizaci CSS pravidel dnes existuje mnoho nástrojů jako je například Unused CSS, uCSS. [49]

Minimalizací rozsahu a hloubky DOM stromu lze docílit taktéž zrychlení. Je doporučeno projít strukturu HTML dokumentu a veškeré prvky, které slouží jako obaly HTML prvků, odstranit. Platí zde jednoduché pravidlo, čím méně prvků bude mít DOM strom, tím rychleji dojde k překreslení stránky jádrem prohlížeče. [49]

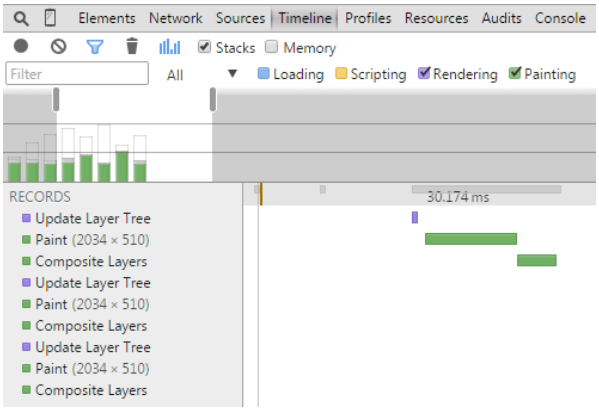
Dále se doporučuje upravovat styly elementů skrze CSS třídy, oproti modifikaci jednotlivých HTML prvků, se totiž při nadefinování obecné třídy a následnému přiřazení více prvků stránka nebo její část překreslí naráz, nikoliv po změně každého HTML prvků zvlášť. Pro příklad, máme CSS třídu *upraveny_element* [49]:

.upraveny_element { width: 100px; height: 200px; margin: 10px; }

Tuto třídu lze pak následně přiřadit všem HTML prvkům pomocí JavaScriptu, u kterých bylo cílem změnit styl. Všechny změny HTML elementů pomocí JavaScriptu, které ovlivňují, jakým způsobem se stránka překreslí, tedy jestli celá či jen část.

3.4.5 Analýza plynulosti vykreslování

Podrobnou analýzu plynulosti vykreslování stránky neboli renderování, lze provést například v prohlížeči Chrome pomocí vývojářské záložky Timeline (od verze Chrome 58 se jmenuje Performance). [46]



Obrázek 29 - Záložka Timeline (od verze Chrome 58 se nazývá Performance) [46]

Rozlišují se různé metriky rychlosti načtení stránek, například od načtení prvního snímku nebo také posledního snímku. Ovšem tyto metriky jsou především dány přenosovou rychlostí a velikostí dat. U vykreslování jsou ovšem výše zmíněné interakce (faktory), které můžou výrazně zpomalit vykreslení či překreslení stránky. [46]

Za ideální stav lze označit hodnotu framerate, tedy počet snímků, kdy jeho hodnota zůstává nad 60 FPS (frames per second – snímek za sekundu). Tato hodnota je totiž většinou obnovovací frekvencí většiny obrazovek. Tento nástroj je velmi dobře použitelný pro analýzu překreslovaných částí stránek. Ovšem zkoumání snížení hodnoty framerate je zde velmi zdlouhavý proces, kdy webový vývojář musí vypínat jednotlivé části vzhledu. Tím je myšleno například pozadí, fixní pozice a tak dále. [46]

3.5 Drahé CSS vlastnosti

Tento pojem pochází z doslovného překladu anglického výrazu „expensive CSS properties“. Jak již název napovídá, jde o takové CSS vlastnosti, které se vykreslují v prohlížeči delší dobu než jiné. Je to způsobeno hlavně náročností vykreslení, složitostí pozicování jednotlivých elementů a také překreslováním stránky webovým prohlížečem. Tento fakt byl již zmíněn v předchozích kapitolách o překreslování stránky pomocí repaint a reflow. Pro odhalení konkrétních vlastností u zkoumaného webu, lze použít záložku Timeline (jen v Google Chrome). [52], [53], [54]:

Obecně lze říci, že se jedná o všechny CSS vlastnosti, které manipulují s prvky nebo se spoléhají na výpočet určitého rozměru. Často označované jako „drahé“ CSS vlastnosti jsou [55]:

- *border-radius* – tato vlastnost určuje zakulacení rohů prvku,

- *box-shadow* – nastavuje stín blokového prvku,
- *opacity* – určuje průhlednost prvku,
- *transform* – umožňuje prvek různými způsoby transformovat (zkosit, posouvat, natahovat, natáčet),
- *filter* – tato vlastnost umožňuje vytvářet grafické efekty prvku.

3.6 Optimalizace rychlosti načtení

Nejčastější problém většiny webu je pomalé načítání. Je vždy doporučováno myslet na rychlost již při tvorbě webu, ovšem ne vždy lze toto dodržet. Optimalizace je obor, ve kterém webový vývojář musí mít hodně znalostí a také dovednosti. Mezi nejčastěji diskutované problémy a jejich řešení lze uvést například zmenšení datového objemu – zmenšení obrázků, zmenšení webového fontu, zmenšení ostatních externích souborů jako jsou JavaScriptové soubory či CSS styly. [56], [57]

3.6.1 HTTP dotazy

Každá načítaná webová stránka má určitý počet HTTP dotazů. Je to způsobeno nutností načíst externí soubory, které jsou na dané stránce potřebné k vykreslení. Soubory se většinou nacházejí na webovém serveru, odkud jsou pomocí HTTP dotazů v internetovém prohlížeči stahovány. Načtení a vykreslení webu lze zrychlit snížením počtu těchto stahovaných souborů. Na stránce by neměly být soubory, které se vůbec nepoužívají. Jedna z cest je přehodnocení stávajícího designu – tedy udělat nový design jednodušší a zároveň optimalizovaný pro rychlejší načítání. Používání hlavně CSS nikoliv velkého počtu obrázku a jiné. Ne vždy tato možnost ovšem jde udělat, a tak existuje i další cesta – pospojovat stejné typy souborů do jednoho. Těmi jsou myšleny hlavně JavaScriptové a CSS soubory. Spojením totiž vznikne pouze jeden dotaz namísto několika předchozích. [56], [57], [58]

Dále také můžeme využít „sprite“ soubory. Tyto sprite soubory obsahují více obrázků, na které je pak pomocí CSS vlastnosti *background-position* odkazováno. Výsledek je takový, že se stahuje pouze jeden obrázek – tedy jeden HTTP dotaz. Také lze využít tzv. data URI, tedy protokolu data, který umožňuje vkládat obrázky do webové stránky. Je zde ovšem problém v tom, že vloženými obrázky nabývá velikost stránky. [59], [60]

Protokol HTTP, který velkou řadu let zůstal nezměněn, se dočkala nové verze. Velká většina těch, co řeší rychlost mají HTTP/2 již nasazené. Druhou verzi protokolu HTTP používá již 40,7 % webových stránek. Díky HTTP/2 je minimalizování HTTP požadavků zbytečné. Jelikož protokol HTTP/2 využívá tzv. multiplexing – dovolí odeslat a přijmout prohlížeči více požadavku najednou v jednom spojení. [23], [61]

3.6.2 Použití CDN

O Content Delivery Network již bylo zmíněno výše, jedná se o síť pro doručování obsahu. Web a celý jeho obsah tedy není uložen pouze na jednom serveru, ale je na více serverech po celém světě. CDN se používá hlavně kvůli zlepšení času odezvy serveru a větší rychlosti. Tato technologie se vyplatí především webům, které navštěvují lidé z celého světa. [56], [62]

3.6.3 Přidání expirace souborů

Internetové prohlížeče používají keš pro snížení počtu požadavků, což dělá web rychlejším. Webový server používá expirující hlavičky pro sdělení klientovi (internetovému prohlížeči), že již stažený soubor má nějakou dobu platnosti. Po uplynutí této platnosti se soubor stahuje znovu. Kešování se používá nejenom pro soubory typu obrázky, ale i pro JavaScriptové soubory a CSS styly. Jako další alternativa k expiraci je používat Max-Age, což je metoda představená v HTTP/1.1. Tato metoda navazuje na expirace v hlavičkách uváděním v hlavičce *Cache Control: max-age.* [57], [58]

3.6.4 Použití Gzip

Metoda Gzip je jedním ze způsobů, jak zmenšit datový objem načítaného obsahu. Gzip funguje na principu, kdy webový server je schopný zabalit data a na straně klienta – tedy internetový prohlížeč je zase schopný si data rozbalit. Gzip přišel s verzí HTTP/1.1, již ho využívá celkem 78,5 % webů. [58], [63]

Library	Size	Compressed size	Compression ratio
jquery-1.11.0.js	276 KB	82 KB	70%
jquery-1.11.0.min.js	94 KB	33 KB	65%
angular-1.2.15.js	729 KB	182 KB	75%
angular-1.2.15.min.js	101 KB	37 KB	63%
bootstrap-3.1.1.css	118 KB	18 KB	85%
bootstrap-3.1.1.min.css	98 KB	17 KB	83%
foundation-5.css	186 KB	22 KB	88%
foundation-5.min.css	146 KB	18 KB	88%

Obrázek 30 - Srovnání Gzip komprese oproti normální velikosti knihoven [64]

3.6.5 Optimalizace obrázků

Dnes jsou již velmi podstatnou součástí webových stránek. Je ovšem velmi nutné optimalizovat jejich velikost. Jedná se totiž o nejčastěji stahovanou položku na webu. Je také velmi podstatné vybrat ten správný formát. Google radí využívat novější webové formáty jako je WebP. Existuje mnoho nástrojů pro optimalizaci obrázků, jako je ImageOptim. Tento nástroj ovšem není automatizovaný proto již v roce 2017 vzniklo mnoho nástrojů, které nabízejí zautomatizování této optimalizace jako je například imagemin. Většina CDN služeb již umí optimalizovat obrázky také. [65]

3.6.6 Minimalizovat DOM

Jednou z velmi účinných metod, jak zrychlit načtení stránky je minimalizací DOM stromu. Minimalizace nezrychlí jen načtení ale i následné operace nad stromem. DOM elementem se zde rozumí všechny HTML tagy, které stránka obsahuje. Zásadní problém ovšem nezpůsobuje jen samotný DOM strom, ale i následný zásah skriptu, který upravuje prvky DOM stromu. Je zde velmi markantní rozdíl, pokud skript vykonává iteraci nad 400 či 4000 elementy DOM stromu. Platí zde pravidlo, čím menší DOM strom, tím rychlejší načtení. Toto pravidlo vyplývá z principu vykreslení stránky pomocí prohlížeče. Google ve svých příručkách pro webové vývojáře radí, aby DOM strom měl méně jak 1500 prvků. Další rada je, aby skript, který je psaný pro úpravu HTML byl psaný velmi specificky. Jedná se o vyhnutí se obecných selektorů typu *, div či pomocí názvu třídy, která je pak následně na stránce použita například tisíc krát. [49], [56], [58]

3.6.7 Načítání JavaScriptu

JavaScriptové soubory blokují vykreslování stránky. Je to způsobeno tím, že prohlížeč nedokáže zobrazit stránku, dokud nestáhnou všechen JavaScript, který není označen jako *async* či *defer*. Pomocí těchto dvou parametrů lze prohlížeči říci, že nemusí čekat na stažení a může vykreslit stránku. Dále je důležité rozmyslet, zda veškeré JavaScriptové soubory mají být v tagu hlavičky. Lze totiž tyto externí soubory umístit až za kritickou část – tedy tu kterou návštěvník vidí první nebo až úplně na konec stránky. [20], [57], [58]

3.6.8 Lazy loading

Lazy loading neboli „líné načítání“ je technika, která načítá datově větší prvky stránky až po načtení celé stránky. Tato technika lze použít jak pro JavaScriptové knihovny, tak pro vkládané prvky jako jsou iframe, videa například z YouTube či pro větší počet obrázků. Také lze použít pro větší a složitější struktury DOM. [39]

Od verze Chrome 76 lze toto načítání definovat pomocí HTML atributu *loading=“lazy“*, lze aplikovat nejenom na obrázky ale také na iframe. Líné načítání lze také provést pomocí knihoven jako jsou LazyLoad , lazySizes či Unveil. [66]

3.6.9 Použití Webpack

Mnoho moderních webových aplikací často pro svojí produkční verzi používají tzv. sdružovací nástroj (tj. z angl. bundling tool). Jedním z těchto nástrojů je například Webpack. Webpack funguje na principu tzv. rozdělení kódu (tj. z angl. code-splitting). To znamená, že již existující kód je rozdělen do optimalizovaných částí, které jsou používány na různých stránkách. Pak na části, které jsou použité pouze na jedné stránce a dále také na kód, který není použitý vůbec. Jednotlivé části rozděleného kódu se tak používají na dané potřebě stránky a dochází tak k velmi dobré optimalizaci načítaných souborů. V daný okamžik klient stahuje, načítá a vykresluje pouze potřebné pro určitou stránku, na které se nachází. [67], [68]

4 Vlastní práce

4.1 Vzorové webové stránky

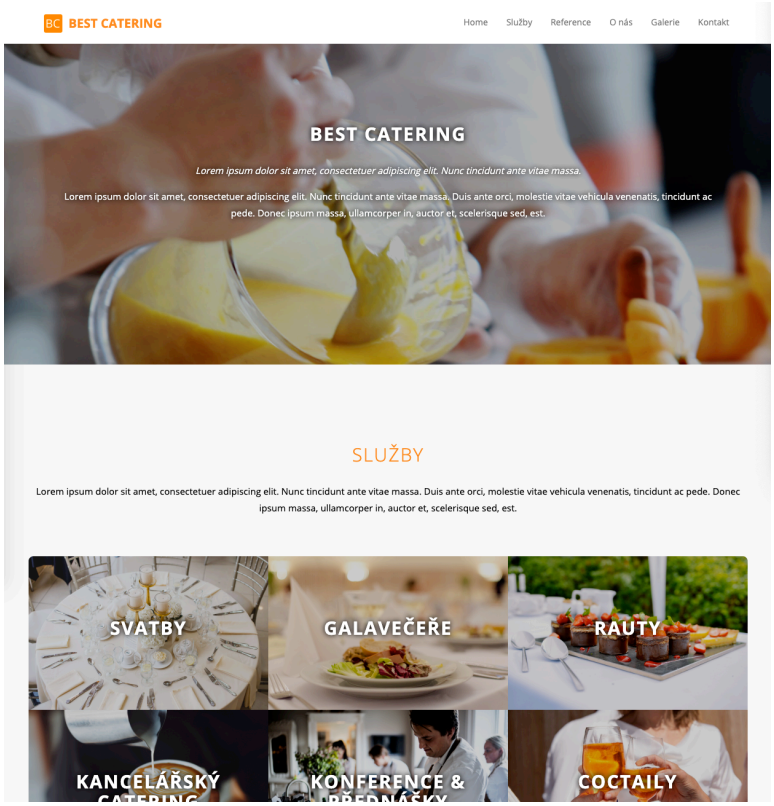
Pro praktickou část této práce byly vytvořeny vzorové webové stránky, které vychází z reálného použití. Informace zde uvedené, veškerý obsah i značka (tj. název) byly změněny. Stránky byly vytvořeny také na základě poznatků z teoretické části a již známých souvislostí. Slouží především ke srovnání rychlosti načtení webu, při použití různých přístupů organizace kaskádových stylů. Měření bylo provedeno pro každou organizaci pětkrát s tím, že odlehlé a extrémní hodnoty byly odstraněny, za pomoci dodatečného změření.

Dále byla vzorová stránka změřena pomocí nástrojů jako jsou Webpagetest.org a PageSpeed Insight. Na základě navrhnutých úprav uvedenými nástroji byly tyto změny aplikovány na vytvořený vzorový web, aby došlo k zrychlení načtení a vykreslení vzorové webové stránky.

4.1.1 Tématika a velikost webových stránek

Stránky byly vytvořeny pro fiktivní podnikatelský subjekt, jehož obor podnikání je gastronomie, konkrétně se jedná o společnost, zaměřenou na pořádání svateb, narozeninových oslav, firemních večírků a ve svém portfoliu má veškeré cateringové služby s tím související. Podnikatelský subjekt je pojmenován jako Best Catering.

Pro účely měření byla vytvořena jednodušší webová prezentace, která je typická především pro menší rodinné podniky. Tento web slouží k prezentaci dané společnosti, jejímu předmětu podnikání a zkušenostem, dále také nabízeným službám. Nejedná se o žádný složitý webový portál nebo informační systém.



Obrázek 31 - Náhled vzorových webových stránek

4.1.2 Použité technologie

Webová stránka je umístěna na webhostingu od společnosti Forpsi. Hostingová služba „EASY“ je ideální k vytvoření webové prezentace společností, které nemusí řešit příliš velkou návštěvnost a jejich existence má za primární účel prezentaci možností dané firmy. Slouží tedy k poskytnutí referencí a ukázek práce. Hosting má neomezený prostor pro soubory, což je výhoda v případě potřeby nahrání fotografií i jiných souborů ve vysoké kvalitě.

Webový hosting používá operační systém Linux. V prostředí Linuxu běží webový server Apache verze 2.2. Webový server Apache je dodáván pod licencí Apache Foundation. Na webovém hostingu je verze PHP 7.3. Webové stránky používají technologie PHP, HTML a CSS. Backendová část webu používá PHP pro jednodušší správu a editaci. Fontendová část webové stránky používá technologie HTML a CSS.

Jednotlivé šablony jsou organizovány do PHP souborů, které jsou dále pro jednoduchost a přehlednost vkládány přímo do stránky. Dochází tak k postupnému složení webové stránky pomocí jednotlivých malých samostatných celků. Tento přístup má

výhodu v jednoduchosti, přehlednosti a velmi dobré udržitelnosti kódu, jako takového. V jednotlivých šablonách je používána technologie HTML.

Na webových stránkách se neobjevuje pouze HTML a CSS, v malé míře je zde používán i JavaScript. Veškerý JavaScript je načítán až na samém konci stránky, což se projeví efektem, kdy stránka nečeká na načtení daných skriptů. Do stránek je aplikován JavaScript, který přidává vhodnou funkcionalitu – po kliknutí na obrázky se zobrazí otevírací galerie. Pro správnou funkčnost této malé knihovny musí být načtena také JavaScriptová knihovna jQuery.

4.1.3 Měřená stránka

Hlavní stránka webové prezentace zvoleného podnikatelského subjektu je skládána z jednotlivých šablon. Tento způsob předchází duplicitám v kódu a jak již bylo zmíněno – vytváří jednodušší udržitelnost kódu. V případě hlavní stránky existuje soubor s názvem index.php, ve kterém je jednak HTML kód a dále jsou do stránky vkládány jednotlivé bloky dalšího HTML kódu, pomocí PHP funkce include, jako je vyobrazeno na obrázku č.31.

```
<!DOCTYPE html>

<html lang="cs">

<head>

  <title>Best Catering - Home</title>
  <?php include('_head.php'); ?>

</head>

<body id="myPage" data-spy="scroll" data-target=".navbar" data-offset="20">


<?php include('_header.php'); ?>
```

Obrázek 32 - Náhled zdrojového kódu hlavní stránky

V tomto případě lze vidět, že veškerý kód, tedy meta data stránky, který je vložen do hlavičky, se nachází v souboru *_head.php* . V souboru *_header.php* se nachází viditelná část hlavičky webu, která obsahuje logotyp a menu. Dále hlavní stránka obsahuje také soubor *_footer.php*, který obsahuje patičku webu s kontaktními údaji.

Stránka samotná je stylizována jako tzv. single page, kdy je veškerý její hlavní obsah pouze na hlavní stránce - *index.php*. Dále v jednotlivých sekcích existují odkazy na podstránky, s detailnějšími informacemi o dané sekci. Menu webové prezentace tedy nemá

odkazy na podstránky, ale umožňuje návštěvníkovi pomocí kotvy v odkazu, se vertikálně pohybovat po hlavní stránce.

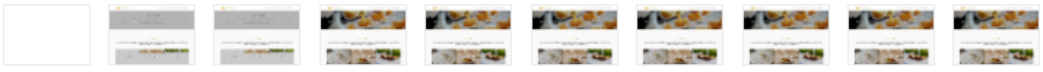
4.2 Srovnání rychlosti načtení různých přístupů organizace CSS

Na základě vytvořených vzorových webových stránek lze vytvořit několik přístupů k organizaci kaskádových stylů a následnému zaměření rychlosti načtení konkrétního webu.

Na vytvořených vzorových stránkách lze zaměřit vytížení procesoru a paměti na straně klienta. Tyto hodnoty jsou zaměřeny pomocí prohlížeče Google Chrome, který nabízí tuto funkcionalitu. Do nástroje se uživatel dostane pomocí menu / Další nástroje / Správce úloh. Vzorový web ve své základní verzi vkládá styly v jednom souboru pro celou stránku. V dalších testováních bude hlavní část rozdělena podle různých přístupů, do několika souborů a ty budou následně připojeny do stránky na různých místech.

4.2.1 Organizace č.1 - Jeden CSS soubor

V tomto případě organizace kaskádových stylů dochází k tomu, že do každé stránky vzorového webu je v hlavičce připojen pouze jeden soubor typu CSS viz kapitola 3.2.7. Všechna CSS pravidla jsou tedy pouze v jediném souboru, v tomto případě v all.css. Daná stránka obsahuje 159 prvků DOM modelu. Tento soubor obsahuje 157 CSS selektorů neboli pravidel pro stylování stránek. Nástroj PageSpeed Insights od společnosti Google je dostupný jako online nástroj pro měření rychlosti a výkonu webu. První měřený způsob uspořádání CSS stylů dosáhl skóre 97 bodů ze 100. Toto hodnocení je udělováno na základě metrik zmíněných níže.



Obrázek 33 - Vizuální zobrazení průběhu načítání webu

Měření bylo provedeno pětkrát, pro získání co nejpřesnějších dat. Podle nástroje, ve kterém je prováděno měření, jsou všechny sledované metriky v normě. Nástroj PageSpeed Insights pomocí provedeného měření navrhuje změny pro optimalizaci načtení a následného vykreslení webových stránek.

Metrika	Měření [s]					Průměr [s]
	č.1	č.2	č.3	č.4	č.5	
První vykreslení obsahu	0,7	0,7	0,7	0,7	0,7	**0,7**
Index rychlosti	1,2	1,1	0,9	1,1	1,2	**1,1**
Doba do interaktivity	1,6	1,6	1	1	1,6	**1,36**
První smysluplné vykreslení	0,7	0,7	0,7	0,7	0,7	**0,7**

Tabulka 1 - Porovnání naměřených hodnot vybraných metrik

Vykreslení	Měření [s]					Průměr [s]
	č.1	č.2	č.3	č.4	č.5	
Rendering	0,86	0,88	0,89	0,7	0,7	**0,806**
Layout	0,06	0,05	0,05	0,06	0,07	**0,058**
Celkem						**0,864**

Tabulka 2 - Porovnání samotného vykreslení webových stránek

Jak lze vidět v tabulce vykreslení webových stránek výše, v průměru došlo k vykreslení stránek za 0,864 sekundy. Dále měření proběhlo v nástroji WebPageSpeed.org, kde byl jako výchozí prohlížeč zvolen Mozilla Firefox. Bohužel tento nástroj nenabízí stejné metriky jako Google PageSpeed Insights, proto budou vybrány pouze metriky, které obsahuje i přechozí měření přes nástroj od společnosti Google.

Dostupné metriky	Měření [s]					Průměr [s]
	č.1	č.2	č.3	č.4	č.5	
První vykreslení obsahu	0,77	0,78	0,57	0,51	0,68	**0,662**
Index rychlosti	0,49	0,4	0,38	0,37	0,37	**0,402**

Tabulka 3 - Porovnání naměřených metrik v nástroji WebPageSpeed.org

Jak lze vidět v tabulce naměřených hodnot jednotlivých metrik, u prvního způsobu došlo ke zlepšení času vykreslení prvního obsahu. Také u metriky „Index rychlosti“ došlo k vylepšení hodnot. Nástroj nenabízí hodnoty Rendering a Layout, které vypovídají o samotném vykreslení stránky. Proto pro další měření bude použit již pouze nástroj Google PageSpeed Insights.

V prohlížeči Chrome přes nástroj Správce úloh, který je vyvinut přímo společností Google a nabízí zobrazení procesů pouze pro daný prohlížeč. Bylo provedeno pět měření a jejich hodnoty byly zprůměrovány. GPU akcelerace je specifická vlastnost prohlížeče Google Chrome, kdy umožňuje zrychlit vykreslování pomocí zapojení grafické karty.

V tomto případě pomocné vlákno GPU vytížilo procesor z 39 %. Dále vlákno GPU zabralo 1,3 GB operační paměti z celkových 8 GB.

	Měření					Průměr
	č.1	č.2	č.3	č.4	č.5	
Obsazená paměť [MB]	464	450	464	471	482	**466,2**
CPU [%]	59,8	77,1	65,6	62	63	**65,5**

Tabulka 4 - Vytížení na straně klienta

Vzorový web s organizací CSS stylů v jednom souboru tedy vytěžuje klienta obsazením paměti RAM ve výši 466,2 MB. Procesor je vytěžován v okamžiku vykreslení stránky v průměru 65,5 % z celkové kapacity. Je nutno zmínit, že prohlížeč Google Chrome umí procesy zpracovat tak, že rozděluje vykreslení webové stránky na několik vláken, tak aby došlo k co nejrychlejšímu vykreslení.

4.2.2 Organizace č.2 - Více CSS souborů

Tento způsob organizace kaskádových stylů spočívá v rozdělení jednoho souboru CSS do několika menších celků. Většinou se tyto menší celky organizují podle určitých komponent či úprav stylů, ovšem nejedná se o komponentové CSS viz kapitola 3.2.8.

```
<link href="general.css" rel="stylesheet" type="text/css">
<link href="grid.css" rel="stylesheet" type="text/css">
<link href="navbar.css" rel="stylesheet" type="text/css">
<link href="content.css" rel="stylesheet" type="text/css">
<link href="footer.css" rel="stylesheet" type="text/css">
```

Obrázek 34 - Napojení CSS stylů do webové stránky

Výše zobrazené soubory jsou připojeny v hlavičce každé jednotlivé HTML stránky webu. V případě vzorových stránek je tento kus kódu vložen pouze do souboru *_head.php*, který je pak dále pomocí PHP funkce *include()* vkládán do všech stránek webu.

Tato organizace umožňuje programátorům či kodérům šablon lepší orientaci a udržitelnost kaskádových stylů, které se používají na daném projektu.

Velikost DOM stromu zůstala od předešlého měření stejná a to je 159 prvků. Celkem došlo k úpravě souborů s CSS pravidly. Nyní se do stránky vkládá 5 souborů: *general.css, grid.css, navbar.css, content.css, footer.css*. Soubor *general.css* obsahuje 49 pravidel pro stylování. Soubor *grid.css* obsahuje 21 pravidel, soubor *navbar.css* obsahuje 42 pravidel. Dále soubor *content.css* obsahuje 40 pravidel a jako poslední soubor *footer.css* obsahuje 5

pravidel. V celkovém součtu je to přesně 157 pravidel, stejně jako v předešlém prvním měření. Měření bylo opět provedeno pětkrát. Na základě měření nástroj PageSpeed Insights vyhodnotil celkové skóre webových stránek na 99 bodů ze 100.

Vybrané metriky	Měření [s]					Průměr [s]
	č.1	č.2	č.3	č.4	č.5	
První vykreslení obsahu	0,7	0,7	0,7	0,7	0,7	**0,7**
Index rychlosti	1	1,1	1.0	1	1	**1,025**
Doba do interaktivity	1,2	1,1	0.9	1,6	1,7	**1,4**
První smysluplné vykreslení	0,7	0,7	0,7	0,7	0,7	**0,7**

Tabulka 5 - Měření vybraných metrik pro organizaci stylů č.2

Z předešlé tabulky můžeme vidět mírné zlepšení u metriky „Index rychlosti“. Naopak u metriky „Doba do interaktivity“ došlo k zvýšení hodnoty na 1,4 sekundy, což je zhoršení o 0,04 sekundy oproti první organizaci CSS stylů. Ostatní metriky jako je „První vykreslení obsahu“ a „První smysluplné vykreslení“ zůstaly stejné, ovšem při vizuálním srovnáním načtení první a druhé organizace CSS stylů můžeme vidět určitý posun k dřívějšímu zobrazení obsahu.

Vykreslení	Měření [s]					Průměr [s]
	č.1	č.2	č.3	č.4	č.5	
Rendering	0,65	0,78	0,67	0,8	0,51	**0,682**
Layout	0,05	0,05	0,05	0,055	0,06	**0,053**
Celkem						**0,735**

Tabulka 6 - Výsledek vykreslení stránek s organizací stylů č.2

Po zaměření samotného vykreslení vzorové webové stránky lze říci, že došlo k rychlejšímu vykreslení. Po výsledném sečtení zprůměrovaných hodnot v rámci pěti měření jsou pořád průměrné hodnoty menší než v první organizaci CSS stylů. Došlo ke zrychlení z 0,864 sekund na 0,735 sekund.

	Měření					Průměr
	č.1	č.2	č.3	č.4	č.5	
Obsazená paměť [MB]	428	425	424	434	429	**428**
CPU [%]	62,5	65,3	57,8	58,2	61,2	**61**

Tabulka 7 - Vytížení na straně klienta

Vykreslování pomocí zapojení grafické karty (tj. řádek GPU) vytížilo procesor z 35 %. Dále vlákno GPU zabralo 1,3 GB operační paměti z celkových 8 GB. Vzorový web

s organizací CSS stylů v do několika menších souborů vytěžuje klienta obsazením paměti RAM ve výši 428 MB. Procesor je vytěžován v okamžiku vykreslení stránky v průměru 61 % z celkové kapacity.

4.2.3 Organizace č.3 - Komponentové CSS

Komponentová organizace kaskádových stylů spočívá v rozdělení jednoho souboru CSS do menších celků, které odpovídají vždy jedné komponentě v dané stránce viz kapitola 3.2.5. Výhodou této možnosti organizace je, že na straně klienta se stahují pouze potřebné CSS soubory pro vykreslení prvků na dané stránce.

V ideálním případě lze tedy říci, že dojde k velké redukci stahovaných dat a také následných stylů, které se aplikují na DOM strom. Jelikož se na dané stránce stahují pouze styly, které jsou nezbytné pro vykreslení stránky, nikoliv všechny styly jako bylo v proběhlém prvním měření, kde styly byly organizovány pouze do jediného souboru.

Komponentové organizovaní stylů je podle internetových zdrojů občas označovaný jako progresivní načítání stylů. Tento způsob spočívá především v načtení stylů těsně před použitím daného kódu. Při komponentové organizaci kaskádových stylů nemusí docházet k redukci duplicit v CSS pravidlech, ale naopak. Díky komponentové orientaci musí každé pravidlo být jasně definováno pro danou komponentu. Tím pádem, pokud se bude na stránce opakovat element , v každé komponentě by správně měl být definován znovu. Dochází ke zlepšení přehlednosti a snadnější udržitelnosti a správě daných stylů. Nedochází pak k zanechání nepoužitých CSS pravidel, jelikož při změně komponenty či jejímu úplnému odstranění ze stránek se rovnou smaže i CSS soubor, který je většinou umístěn na stejné úrovni před daným celkem.

Před dalším měřením jsou soubory s CSS styly rozděleny podle různých komponent, tedy celků, které se nacházejí na webových stránkách. Vzniklo tedy 9 CSS souborů, z toho 2 jsou pro jiné stránky, než je hlavní měřená stránka, tím pádem došlo k ušetření pár CSS pravidel, které se nemusí načítat, pokud nejsou potřeba. Zbylých 7 CSS souborů je různě podle použití daných komponent připojených těsně před danou komponentou. Například soubor navbar.css je připojen hned nad HTML prvkem <*div class=“navbar“*>, který tento soubor používá.

Obrázek 35 - Příklad použití a připojení stylu navbar.css do viditelné hlavičky webu

V provedeném měření 3. způsobu organizace CSS stylů bylo dosaženo celkového skóre podle nástroje PageSpeed Insights 99 bodů ze 100, stejně jako při použití druhého způsobu. Velikost DOM stromu zůstala stejná jako u předchozích měření, tedy na hodnotě 159 prvků.

Vybrané metriky	Měření					Průměr [s]
	č.1 [s]	č.2 [s]	č.3 [s]	č.4 [s]	č.5 [s]	
První vykreslení obsahu	0,6	0,6	0,5	0,6	0,6	**0,58**
Index rychlosti	1,2	1,1	0,9	1,1	0,9	**1,04**
Doba do interaktivity	1,7	1,6	1,7	1,1	1,8	**1,58**
První smysluplné vykreslení	0,7	0,6	0,7	0,7	0,6	**0,66**

Tabulka 8 - Měření vybraných metrik organizace stylů č. 3

Jak lze vidět na proběhlém měření 3. způsobu organizace CSS stylů, u metriky „První vykreslení obsahu“ došlo k dosažení lepších hodnot než v přechozích měřeních. U metriky „Index rychlosti“ došlo naopak k posunu hodnoty z 1,025 na 1,04 – tedy zhoršení. U metriky „První smysluplné vykreslení“ došla také k mírnému zlepšení hodnoty, ovšem pouze o pár setin sekundy u zprůměrované hodnoty.

Vykreslení	Měření [s]					Průměr [s]
	č.1	č.2	č.3	č.4	č.5	
Rendering	1,09	1,15	0,92	0,93	1,28	**1,074**
Layout	0,075	0,056	0,06	0,07	0,075	**0,0672**
Celkem						**1,1412**

Tabulka 9 - Samotné vykreslení stránky s organizací CSS stylů č.3

Samotné vykreslování stránky, které je již očištěné o stahování souborů ve srovnání s předešlým měřením zhoršilo hlavně u samotného vykreslení stránky (tj. Rendering).

V celkovém součtu i s hodnotou překreslení některých částí webu (tj. Layout) došlo k posunu hodnoty z 0,735 až na 1,1412 sekundy.

Lze z tohoto výsledku usoudit, že metriky, které jsou v rámci uživatelského hlediska viditelné, například jako metrika „První vykreslení obsahu“ se zlepšily – tedy uživatel pocítí dřívější reakci na vyvolanou akci, celkové vykreslování stránky se poměrně dost zhoršilo. Je jen a jen otázkou daného webu či klienta, pro kterého je web zhotoven, zda jeho návštěvníci uvítají rychleji zobrazený první viditelný obsah, či uvítají spíše rychleji použitelné stránky (tj. rychlejší vykreslení).

	Měření					Průměr
	č.1	č.2	č.3	č.4	č.5	
Obsazená paměť [MB]	430	429	433	428	426	**429,2**
CPU [%]	61,9	59,7	58,4	54	58,2	**58,44**

Tabulka 10 - Vytížení na straně klienta

Proces GPU vytížil procesor z 34 %. Dále vlákno GPU zabralo 1,2 GB operační paměti z celkových 8 GB.

Vzorový web s organizací CSS stylů v do několika menších souborů vytěžuje klienta obsazením paměti RAM ve výši 428 MB. Procesor je vytěžován v okamžiku vykreslení stránky v průměru 61 % z celkové kapacity.

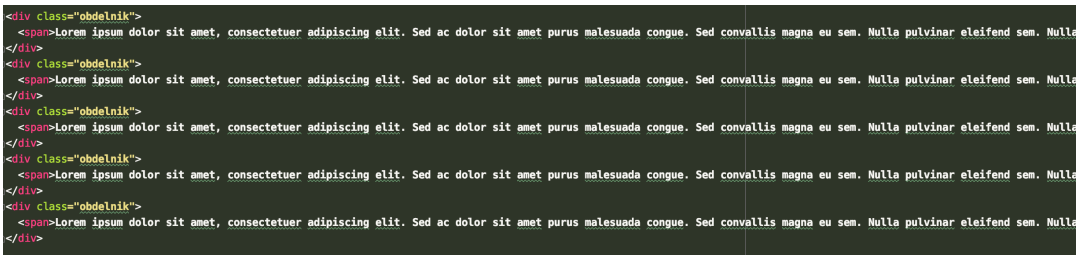
4.3 Drahé CSS vlastnosti

V následující kapitole dojde k zaměření časů vykreslení tzv. drahých kaskádových vlastností. Pro otestování rychlosti načítání jednotlivých vlastností bude použit prohlížeč Google Chrome na straně klienta.

Zkoumané vlastnosti byli vybrány na základě několika zdrojů, které uvádějí, že je lepší se těmto CSS vlastnostem vyhnout nebo je, pokud možno nepoužívat pro velké množství prvků viz kapitola 3.5.

4.3.1 Vlastnost border-radius

Měření rychlosti načtení vlastnosti border-radius probíhá nejdříve bez dané vlastnosti se základním stylováním, které je přítomno i v následujících měření. V prvním měření se tedy na stránce nachází 5 prvků s CSS třídou *.obdelnik*.



Obrázek 36 - Struktura testovaného HTML souboru

Pro změření počátečních hodnot načtení tedy bylo zvoleno následující stylování, které by mělo být bez tzv. drahých CSS vlastností. Styl vypadá následovně:

```
.obdelnik {
  float: left;
  width: 90%;
  margin-top: 40px;
  margin-left: 20px;
  padding: 10px;
  background-color: #888;
  color: #000;
}
```

Toto měření je velmi důležité, jelikož poslouží jako referenční hodnota, se kterou budou následná měření srovnávat. V následujících měřeních bude použita záložka Performace (tj. Výkon) v prohlížeči Google Chrome. V této záložce jsou udávány hodnoty Rendering a Painting. O těchto dvou pojmech již bylo psáno v kapitole o vykreslovacích jádrech prohlížečů v teoretické částí této práce. Po sečtení těchto dvou hodnot lze říci, že se jedná o celkové vykreslení stránky. Do tohoto vykreslení se nezapočítává čas stažení souborů pomocí HTTP dotazu, jelikož soubory jsou na lokálním disku a také proto, že měření probíhá na straně klienta.

4.3.1.1 Bez vlastnosti border-radius

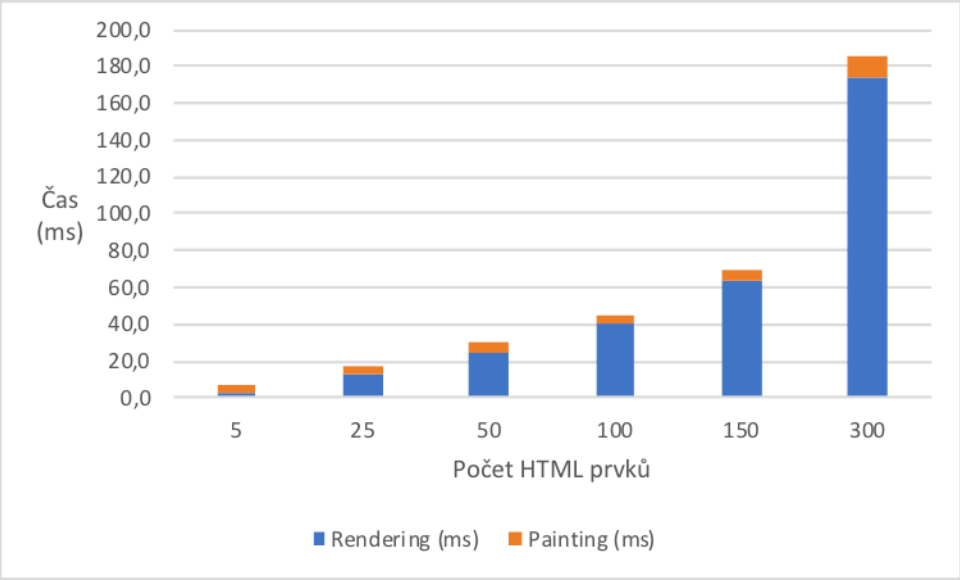
Počet prvků	Rendering [ms]	Painting [ms]	Celkem [ms]
5	3,4	4,2	7,6
25	12,6	4,6	17,2
50	24,4	6	30,4
100	40,2	5,4	45,6
150	64	6	70
300	173,4	11,4	184,8

Tabulka 11 - Vykreslení různých počtů HTML prvků bez vlastnosti border-radius

V měření 5 HTML prvků došlo k průměrné době Renderingu 3,4ms (tj. milisekund). Průměrná hodnota Paintingu je 4,2ms. Výsledná hodnota celkového vykreslení je 8ms. V měření 25 HTML prvků došlo k průměrné době Renderingu 12,6ms, jak můžeme vidět je velký nárůst oproti hodnotě předchozí. Průměrná doba Paintingu je 4,6ms. U této hodnoty nedošlo k výraznému nárůstu. Výsledná hodnota vykreslení je 17,2ms. Tato hodnota zaznamenala velký nárůst.

U měření 50 a 100 HTML prvků bez vlastnosti border-radius došlo ve sloupci Renderingu k mírnému zvýšení. U hodnoty painting dokonce došlo k mírnému poklesu. Samostatné vykreslení se zvětšuje s přibývajícími prvky. Pro 50 prvků je hodnota 30,4ms a pro 100 prvků již 45,6ms.

V měření 150 HTML prvků došlo k průměrné době Renderingu 64 ms. Průměrná doba Paintingu je 6 ms. Výsledná hodnota celkového vykreslení je 70 ms. Je potřeba zmínit, že došlo k nárůstu doby hlavně kvůli zvětšení počtu prvků DOM stromu. V měření 300 HTML prvků došlo k průměrné době Renderingu 173,4 ms. Průměrná doba Paintingu je 11,4 ms. Výsledná hodnota celkového vykreslení tedy činí 184,8 ms. Je zde opětovně vidět, jak ovlivňuje nárůst prvků DOM stromu rychlost vykreslení stránky. Největší narůst zaznamenaly hodnoty ve sloupci Rendering.



Obrázek 37 - Vývoje hodnot sloupce Rendering a Painting

4.3.1.2 S vlastností border-radius

V následujících měřeních bylo do pravidla CSS označeného jako *.obdelnik { ... }* přidána vlastnost *border-radius s hodnotou 20px*. Jedná se o vlastnost, která zaoblí hrany daného prvku. Celé pravidlo tedy vypadá takto:

```
.obdelnik {
  float: left;
  width: 90%;
  margin-top: 40px;
  margin-left: 20px;
  padding: 10px;
  background-color: #888;
  color: #000;
  border-radius: 20px;
}
```

Očekávané chování je, že by následující měření vykreslení prvků *.obdelnik* měla být pomalejší než u prvků, v kterých chyběla CSS vlastnost *border-radius*.

Počet prvků	Rendering [ms]	Painting [ms]	Celkem [ms]
5	4,6	4,8	9,4
25	14	5,2	19,2
50	31,8	7,4	39,2
100	40,2	5,4	45,6
150	75	7,8	82,8
300	187,6	11,6	199,2

Tabulka 12 - Vykreslení různých počtů prvků s vlastností border-radius

V měření 5 prvků došlo k průměrné době Renderingu 4,6ms a Paintingu 4,8ms. Výsledná hodnota celkového vykreslení tedy činí 9,4ms. Pro 25 a 50 HTML prvků se hodnoty ve sloupci Rendering zvětšují skoro dvojnásobně, ovšem ve sloupci Painting tak velký nárůst nenastal. U měření 100 prvků došlo zase ke zvýšení hodnot, akorát hodnota Painting zaznamenala pokles.

V měření 150 prvků došlo k průměrné době Renderingu 75ms a Paintingu 7,8ms. Výsledná hodnota celkového vykreslení tedy činí 126,8 ms. V měření 300 prvků došlo k více jak dvojnásobnému nárůstu hodnoty Rendering. Průměrná doba Paintingu vzrostla na 11,6ms. Výsledná hodnota celkového vykreslení tedy činí 199,2 ms.

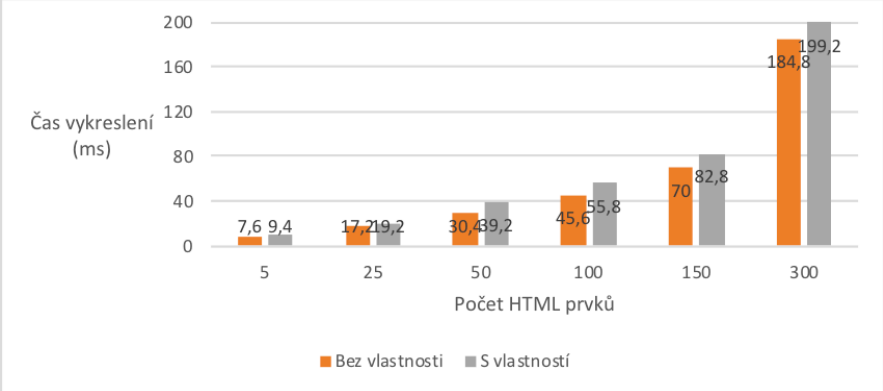
4.3.1.3 Zhodnocení použití vlastnosti border-radius

Jak lze na základě proběhlého měření vidět, podstatným faktorem při vykreslení stránky je již výše zmiňovaný DOM strom. Při srovnání počtu vykreslovaných prvků, pokud se jedná o rozsáhlejší strom, obsahující v měření například 300 prvků, dochází k delšímu času v sloupci „Rendering“. Tento sloupec značí sestavení stromu DOM a následný Painting pak již přiřazuje vzniklému stromu styl a vykreslí stránku přesně na každý pixel.

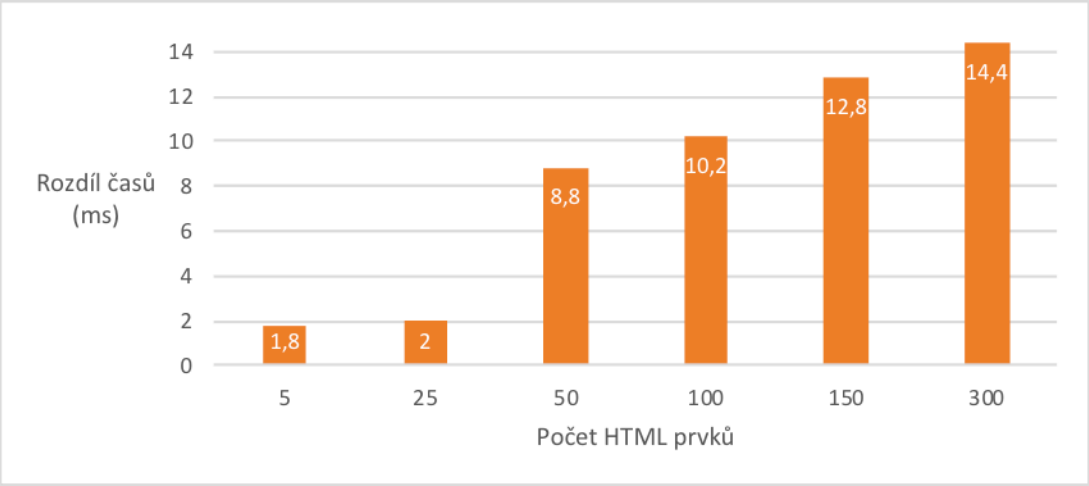
Počet HTML prvků	Doba vykreslení [ms]		Rozdíl časů [ms]
	Bez vlastnosti	S vlastností	
5	7,6	9,4	1,8
25	17,2	19,2	2
50	30,4	39,2	8,8
100	45,6	55,8	10,2
150	70	82,8	12,8
300	184,8	199,2	14,4

Tabulka 13 - Srovnání vykreslení s a bez vlastnosti border-radius

Při srovnání vykreslení stránky s vlastností, respektive bez vlasnosti *border-radius* lze usoudit na základě měření, že tato vlastnost má vliv na čas vykreslení stránky. Při počtu 5 prvků s přiřazeným CSS pravidlem *.obdelnik* je rozdíl pouze 1,8ms. Od 50 HTML prvků se rozdíl rapidně zvedá, proto podle proběhlého měření můžeme říci, že vlastnost border-radius je dobré používat pro menší DOM stromy do 50 prvků, na které se aplikuje pravidlo s danou vlastností. U největšího počtu prvků rozdíl činil 14,4ms. Lze tedy říci, že se zvětšujícím počtem prvků se také zvětšuje rozdíl mezi hodnotami bez a s vlastnotí *border-radius*.



Obrázek 38 - Srovnání vykreslení bez a s vlastností border-radius



Obrázek 39 - Rozdíl časů vykreslení s a bez vlastnosti border-radius

4.3.2 Vlastnost box-shadow

Vlastnost *box-shadow* je používána především pro přidělání stínu pod daným prvkem. Jako první parametr se uvádí barva stínu, pak odsazení na ose X, dále odsazení na ose Y a jako poslední míra rozostření v pixelech. Následující měření bylo provedeno pro 5, 25, 50, 100, 150 a 300 stejných HTML tagů * *. Nejdříve budou změřená rychlost vykreslení bez vlastnosti *box-shadow* a následně s vlastností *box-shadow*.

4.3.2.1 Bez vlastnosti box-shadow

Pro měření byla použita stejná struktura HTML stránky jako je na obrázku č. 39. Styl zůstal pro počáteční měření bez vlastnosti box-shadow také stejný.

Počet prvků	Rendering [ms]	Painting [ms]	Celkem [ms]
5	3,2	3,6	6,8
25	14	4,8	18,8
50	24	5,8	29,8
100	38,4	6,2	44,6
150	63,8	9	72,8
300	130	13,8	143,8

Tabulka 14 - Měření různých počtů prvků bez vlastnosti box-shadow

V měření 5 prvků došlo k průměrné době Renderingu 3,2 ms. Průměrná doba Paintingu je 3,6 ms. Výsledná hodnota celkového vykreslení tedy činí 6,8 ms. Při měření 25 a 50 prvků byl zaznamenán ve sloupci Rendering nárůst hodnot, stejně tak u sloupce

Painting. Celkové vykreslení je pro 25 prvků 18,8 ms a pro 50 prvků 29,8 ms. Měření 100 prvků dosáhlo při Renderingu hodnoty 38,4 ms a při Paintingu 6,2 ms. Výsledné vykreslení proběhlo za 44,6 ms.

V měření 150 prvků došlo k průměrné době Renderingu 63,8 ms. Průměrná doba Paintingu je 9 ms. Výsledná hodnota celkového vykreslení tedy činí 72,8 ms. V měření 300 prvků došlo k průměrné době Renderingu 130 ms. Průměrná doba Paintingu je 13,8 ms. Výsledná hodnota celkového vykreslení tedy činí 143,8 ms.

4.3.2.2 S vlastností box-shadow

Pro měření rychlosti vykreslení struktura dokumentu HTML zůstala stejná, liší se pouze v počtu prvků, podle toho, zda dochází k měření 5, 25, 100, 150 či 300 prvků.

CSS pravidlo *.obdelnik* bylo upraveno. Byla přidána vlastnost *box-shadow*. U této vlastnosti se nejdříve zadává barva stínu, pak odsazení X, dále odsazení Y a jako poslední se uvádí míra rozmazání stínu v pixelech.

```
.obdelnik {
  float: left;
  width: 90%;
  margin-top: 40px;
  margin-left: 20px;
  padding: 10px;
  background-color: #888;
  color: #000;
  box-shadow: #000 0 0 10px;
}
```

Počet prvků	Rendering [ms]	Painting [ms]	Celkem [ms]
5	4,6	3,6	8,2
25	15,4	5,4	20,8
50	23,6	10,6	34,2
100	44,2	7	51,2
150	72,4	8,6	81
300	142,8	12,4	155,2

Tabulka 15 - Výsledky měření s vlastností box-shadow

V měření 5 HTML prvků došlo k průměrné době Renderingu 4,6 ms. Průměrná doba Paintingu je 3,6 ms. Výsledná hodnota celkového vykreslení tedy činí 8,2 ms. Měření 25 a 50 prvků ukázalo, že časový údaj u sloupce Rendering, Painting a celkové vykreslení narůstá se zvyšováním počtu prvků DOM stromu. U měření 100 prvků došlo u sloupce

Rendering k nárůstu hodnoty na 44,2 ms. Ovšem u sloupce Painting došlo ke snížení. Nejspíše se bude jednat o odchylku v měření. Celkové vykreslení vzrostlo na hodnotu 51,2 ms.

Stejně tomu tak je u měření 150 a 300 prvků. V měření 150 prvků došlo k průměrné době Renderingu 72,4 ms. Průměrná doba Paintingu je 8,6 ms. Výsledná hodnota celkového vykreslení tedy činí 116,8 ms. V měření 300 prvků došlo k průměrné době Renderingu 240,2 ms. Průměrná doba Paintingu je 16,2 ms. Výsledná hodnota celkového vykreslení tedy činí 256,4 ms.

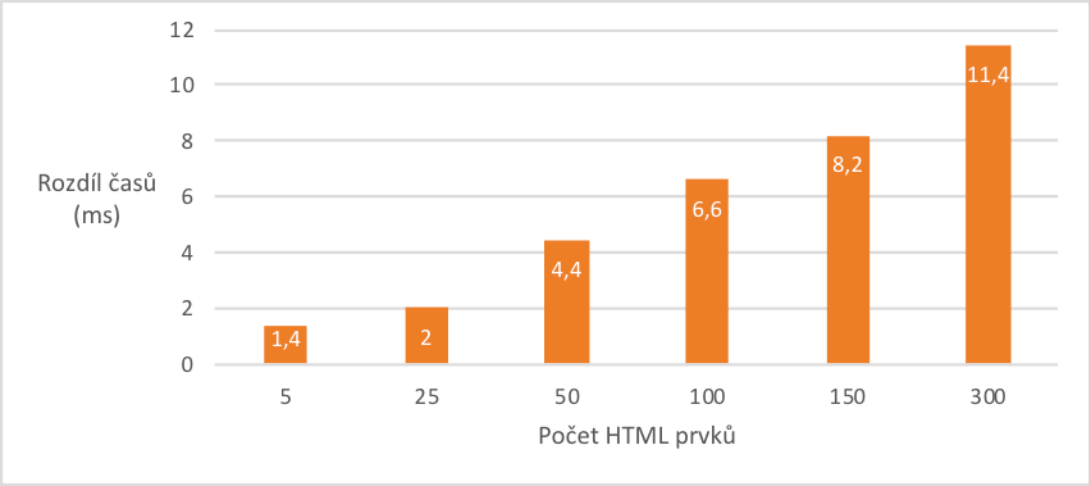
4.3.2.3 Zhodnocení použití vlastnosti box-shadow

Při srovnání rychlosti vykreslení stránky bez vlastnosti box-shadow, respektive s vlastnotí lze říci, že vlastnost box-shadow má vliv na rychlost vykreslení. Ovšem největší rozdíl v časech je stejně jako u předchozí vlastnosti vidět u rozdílných počtů vykreslovaných prvků. Lze tedy usoudit, že velký vliv na rychlost vykreslení má velikost DOM stromu.

Počet HTML prvků	Doba vykreslení [ms]		Rozdíl časů
	Bez vlastnosti	S vlastností	
5	6,8	8,2	1,4
25	18,8	20,8	2
50	29,8	34,2	4,4
100	44,6	51,2	6,6
150	72,8	81	8,2
300	143,8	155,2	11,4

Tabulka 16 - Srovnání časů vykreslení s a bez vlastnosti box-shadow

Rozdíl v měření s a bez vlastnosti pro 5 HTML prvků je 1,4 ms, což je velmi málo znatelný rozdíl. Stejně tak je tomu u počtu prvků 25 a 50. Od počtu 50 a výše dochází k větším rozdílům. Nejmarkantnější rozdíl byl naměřen u 300 HTML prvků, který činí 11,4 milisekundy. Lze proto říci, že tato vlastnost by se mělo aplikovat maximálně do počtu 50.



Obrázek 40 - Srovnání rozdílů s a bez vlastnosti u různých počtů prvků

4.3.3 Vlastnost *opacity*

Vlastnost *opacity* je používána pro zprůhlednění daného HTML prvku, na které se CSS pravidlo s vlastností *opacity* aplikuje. Vlastnost má pouze jeden parametr a tím je číslo z intervalu od 0 do 1.

Pro následující měření bylo vyhotoveno 5 stránek HTML, které obsahují 5, 25, 50, 100, 150 a 300 stejných HTML tagů * *. Nad těmito stránkami proběhne následující měření. Nejdříve bude změřená rychlost vykreslení bez vlastnosti *opacity* a následně s danou vlastností pro které je vytvořeno dalších 5 obdobných stránek.

4.3.3.1 Bez vlastnosti *opacity*

V této kapitole CSS pravidlo *.obdelnik* neobsahuje vlastnost *opacity*. Pro následující měření bylo upraveno CSS pravidlo z předchozích kapitol následovně:

```
.obdelnik {
  float: left;
  width: 90%;
  margin-top: 40px;
  margin-left: 20px;
  padding: 10px;
  background-color: #B70A0D;
  color: #FFF;
}
```

Počet prvků	Rendering [ms]	Painting [ms]	Celkem [ms]
5	4,2	2,8	7,0
25	13	4,4	17,4
50	21,4	6	27,4
100	42,2	6,4	48,6
150	58,6	5,4	64
300	152	9,4	161,4

Tabulka 17 - Výsledky měření bez vlastnosti opacity

V měření 5 prvků došlo k průměrné době Renderingu 4,2 ms. Průměrná doba Paintingu je 2,8 ms. Výsledná hodnota celkového vykreslení tedy činí 7 ms. V měření 25 a 50 prvků došlo k zvýšení hodnot u sloupce Rendering a Painting. Výsledné vykreslení je pro 25 prvků 17,4 ms a pro 50 prvků 27,4 ms.

Měření pro 100 prvků zaznamenalo u hodnoty Rendering skoro dvojnásobný růst, u hodnoty Painting růst pouze o 0,4 ms. V měření 150 prvků došlo k průměrné době Renderingu 58,6 ms. Průměrná doba Paintingu je 5,4 ms. Výsledná hodnota celkového vykreslení tedy činí 64 ms. V měření 300 prvků došlo ke skokovému nárůstu hodnoty Renderingu na 152 ms. Průměrná doba Paintingu se zvýšila oproti přechozí o 4 ms. Výsledná hodnota celkového vykreslení tedy činí 161,4 ms. Oproti předchozí hodnotě u 150 prvků se jedná o zvýšení hodnoty více jak dvojnásob.

4.3.3.2 S vlastností *opacity*

CSS pravidlo *.obdelnik* bylo upraveno. Byla přidána vlastnost *opacity*. U této vlastnosti se zadává číselná hodnota v intervalu od 0 do 1. Pro následující měření byla přidána vlastnost *opacity* s hodnotou 0,2.

```
.obdelnik {
  float: left;
  width: 90%;
  margin-top: 40px;
  margin-left: 20px;
  padding: 10px;
  background-color: #B70A0D;
  color: #FFF;
  opacity: 0.2;
}
```

Počet prvků	Rendering [ms]	Painting [ms]	Celkem [ms]
5	4,4	3,8	8,2
25	14,4	4,8	19,2
50	24,4	5,8	30,2
100	46,4	7	53,4
150	64,6	6,4	71
300	159,4	13,6	173

Tabulka 18 - Výsledky měření s vlastností opacity

V měření 5 HTML prvků došlo k průměrné době Renderingu 4,4 ms. Průměrná doba Paintingu je 3,8 ms. Výsledná hodnota celkového vykreslení tedy činí 8,2 ms. V měření 25 a 50 prvků došlo k nárůstu hodnot Rendering a Painting. Výsledné vykreslení u 25 prvků tedy činí 19,2 ms a u 50 prvků činí 30,2 ms. U měření 100 prvků došlo také k nárůstu hodnot. Výsledné vykreslení činí 53,4 ms.

V měření 150 prvků došlo k průměrné době Renderingu 64,6 ms. Průměrná doba Paintingu je 6,4 ms. Výsledná hodnota celkového vykreslení tedy činí 71 ms. V měření 300 prvků došlo k více jak dvojnásobnému nárůstu hodnoty Renderingu na 159,4 ms. U hodnoty Paintingu došlo také k více jak dvojnásobnému nárůstu. Výsledná hodnota celkového vykreslení tedy činí 173 ms. Oproti 150 prvkům jde o více jak dvojnásobný nárůst.

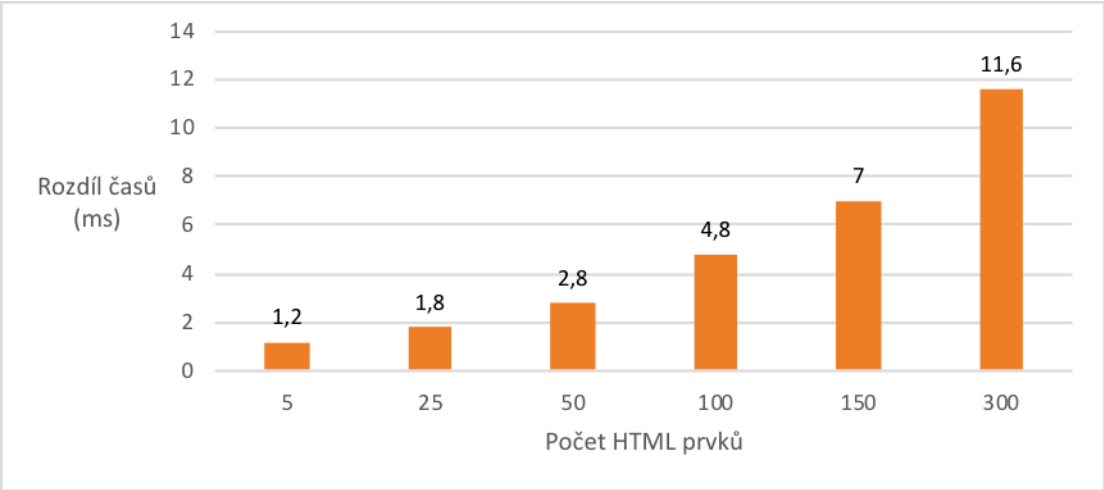
4.3.3.3 Zhodnocení vlastnosti *opacity*

Při srovnání rychlosti vykreslení stránky bez vlastnosti *opacity*, respektive s vlastností lze říci, že tato vlastnost má vliv na rychlost vykreslení, ovšem nelze říci od jakého počtu prvků se nevyplatí tuto vlastnost aplikovat.

Počet HTML prvků	Doba vykreslení [ms]		Rozdíl časů
	Bez vlastnosti	S vlastností	
5	7	8,2	1,2
25	17,4	19,2	1,8
50	27,4	30,2	2,8
100	48,6	53,4	4,8
150	64	71	7
300	161,4	173	11,6

Tabulka 19 - Srovnání naměřených hodnot bez a s vlastností opacity

Lze tedy usoudit, díky velkému přírůstku v hodnotách u 5 různých počtů prvků, že velký vliv na rychlost vykreslení má určitě velikost DOM stromu. K velkému rozdílu v měření s a bez vlastnosti nedošlo. Nelze ani přesně říci, od jakého počtu HTML prvků, na které se aplikuje vlastnost, je nevýhodné pro rychlost načtení. Největší rozdíl byl naměřen u 300 prvků, který činí 11,6 milisekundy.



Obrázek 41 - Rozdíl ve vykreslení s a bez vlastnosti opacity

4.3.4 Vlastnost transform

Vlastnost *transform* je používána pro libovolné transformace daného HTML prvku, na které se CSS pravidlo s vlastností *tranform* aplikuje. HTML prvek můžeme pomocí funkcí, které musíme uvést do vlastnosti, můžeme:

- otáčet - tj. *rotate (stupně),*
- zkosit - tj. skew (X stupně zkosení na ose X, stupně zkosení na ose Y),
- posouvat – tj. translate (posunutí na ose X, posunutí na ose Y),
- natahovat – tj. scale (natažení na ose X, natažení na ose Y).
- Pro každou tranformaci prvku je nutné funkci uvést v dané vlastnosti. Uvedené funkce mají další různé variace. Vlastnost *transform* přišla s verzí CSS 3.

Pro následující měření bylo vyhotoveno 5 HTML stránek, které obsahují 5, 25, 50, 100, 150 a 300 stejných HTML tagů * *. Nad těmito stránkami proběhne následující měření. Nejdříve bude změřená rychlost vykreslení bez vlastnosti dané vlastnosti a následně s vlastností, pro které jsou vytvořeny další obdobné stránky s pozměněným CSS pravidlem *.obdelnik*.

4.3.4.1 Bez vlastnosti *transform*

V této kapitole CSS pravidlo.obdelnik neobsahuje vlastnost *transform*. Pro následující měření bylo převzato CSS pravidlo .obdelnik z předchozí kapitoly.

```
.obdelnik {
  float: left;
  width: 90%;
  margin-top: 40px;
  margin-left: 20px;
  padding: 10px;
  background-color: #B70A0D;
  color: #FFF;
}
```

Počet prvků	Rendering [ms]	Painting [ms]	Celkem [ms]
5	5,2	3,8	9,0
25	13,6	3,2	16,8
50	22	6	28
100	44,8	5	49,8
150	67,4	6	73,4
300	153,6	9,2	162,8

Tabulka 20 - Výsledek měření bez vlastnosti transform

V měření 5 HTML prvků došlo k průměrné době Renderingu 5,2 ms. Průměrná doba Paintingu je 3,8 ms. Výsledná hodnota celkového vykreslení tedy činí 9 ms. V měření 25 a 50 prvků dosáhlo u hodnot Rendering a Painting nárůstu. Také u 100 prvků došlo k nárůstu hodnot oproti předchozím.

V měření 150 prvků došlo k průměrné době Renderingu 67,4 ms. Průměrná doba Paintingu je 6 ms. Výsledná hodnota celkového vykreslení tedy činí 73,4 ms. V měření 300 prvků došlo k průměrné době Renderingu 153,6 ms. Průměrná doba Paintingu je 9,2 ms. Výsledná hodnota celkového vykreslení tedy činí 162,8 ms. Poslední počet dosáhl více jak dvojnásobného nárůstu.

4.3.4.2 S vlastností *transform*

Pro následující měření bylo upraveno CSS pravidlo .obdelnik. Do pravidla byla přidána vlastnot *transform: translate(20px, 40px, 60px) rotate(20deg);*. Pro demonstraci rychlosti vykreslení s vlastností *transform* byly vybrány funkce transformace jako je posun (tj. translate) a otočení (tj. rotate). CSS pravidlo vypadá následovně:

```css
.obdelnik {
  float: left;
  width: 90%;
  margin-top: 40px;
  margin-left: 20px;
  padding: 10px;
  background-color: #B70A0D;
  color: #FFF;
  transform: translate3d(20px, 40px, 60px) rotate(20deg);
}
```

Počet prvků	Rendering [ms]	Painting [ms]	Celkem [ms]
5	6,6	3,8	10,4
25	13,8	5,4	19,2
50	23,2	9,8	33
100	50	8	58
150	74	11,2	85,2
300	161,6	13,6	175,2

Tabulka 21 - Výsledek měření s vlastností transform

V měření 5 prvků došlo k průměrné době Renderingu 6,6 ms. Průměrná doba Paintingu je 3,8 ms. Výsledná hodnota celkového vykreslení tedy činí 10,4 ms. V měření 25 a 50 prvků došlo u hodnot Rendering a Painting k nárůstu. Výsledná hodnota vykreslení pro 25 prvků činí 19,2 ms a pro 50 prvků činí 33 ms.

V měření 100 prvků došlo k předpokládanému vývoji hodnot – tedy k nárůstu. V měření 150 prvků došlo k průměrné době Renderingu 74 ms. Průměrná doba Paintingu je 11,2 ms. Výsledná hodnota celkového vykreslení tedy činí 85,2 ms. Nejvyšších hodnot bylo dosaženo u 300 HTML prvků. Průměrná doba Renderingu je 161,6 ms. Průměrná doba Paintingu je 13,6 ms. Výsledná hodnota celkového vykreslení tedy činí 175,2 ms.

4.3.4.3 Zhodnocení vlastnosti *transform*

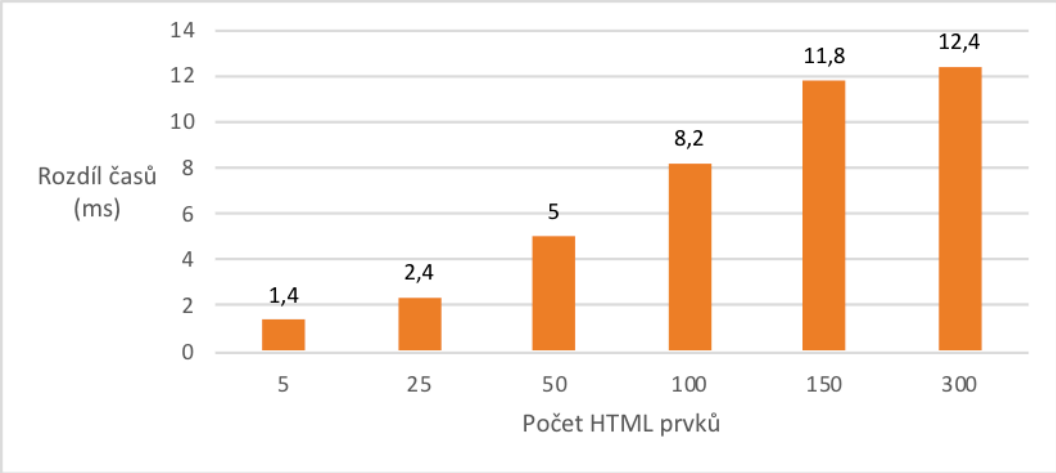
Při srovnání rychlosti vykreslení stránky bez vlastnosti *transform*, respektive s vlastností lze říci, že tato vlastnost má vliv na rychlost vykreslení.

Počet HTML prvků	Doba vykreslení [ms]		Rozdíl časů
	Bez vlastnosti	S vlastností	
5	9	10,4	1,4
25	16,8	19,2	2,4
50	28	33	5
100	49,8	58	8,2
150	73,4	85,2	11,8

300	162,8	175,2	12,4

Tabulka 22 - Srovnání vykreslení bez a s vlastností transform

Při srovnání časů mezi různými počty prvků je vidět, že velký vliv na čas vykreslení stránky má velikost DOM stromu. Rozdíl v měření s a bez vlastnosti pro 5 prvků je 1,4 milisekund. Od počtu 25 prvků se hodnota zvýšila dvojnásobně, proto lze říci, že vlastnost transform je lepší používat do počtu 25 prvků. Pro 150 prvků je tento rozdíl 11,8 milisekund. Největší rozdíl byl naměřen u 300 prvků, který činil 12,4 milisekund.



Obrázek 42 - Rozdíl časů vykreslení bez a s vlastností transform

4.4 Optimalizace vzorových stránek

Pro optimalizaci byl použit první způsob organizace CSS stylů. Optimalizace proběhne na základě výstupů z nástroje Google PageSpeed Insights, který umí webovým vývojářům poskytnout informace o jednotlivých položkách, které brzdí či oddalují nejrychlejší vykreslení webových stránek.

Prvním krokem je zaměřit se na obrázky použité na webové stránce. Jelikož při prvním testování nástroj PageSpeed Insights vyhodnotil, že pokud dojde ke zmenšení velikosti obrázků vložených do webových stránek, dojde i ke zrychlení vykreslení.

▲ **Předejděte přenášení enormního množství dat** — **Celková velikost byla 7 176 kB**

Přenášení velkého množství dat po síti je pro uživatele finančně nákladné a obvykle vede k pomalému načítání. Další informace

URL	Velikost
/img/coffe.jpg (jirikastanek.com)	1 900 KB
/img/catering7.jpg (jirikastanek.com)	705 KB
/img/svatby.jpg (jirikastanek.com)	685 KB
/img/palac-kultury.jpg (jirikastanek.com)	534 KB
/img/kulturni-dum-smichov.jpg (jirikastanek.com)	527 KB
/img/sluzby.jpg (jirikastanek.com)	522 KB
/img/catering6.jpg (jirikastanek.com)	413 KB
/img/serve.jpg (jirikastanek.com)	363 KB
/img/catering5.jpg (jirikastanek.com)	350 KB
/img/gala-evening.jpg (jirikastanek.com)	266 KB

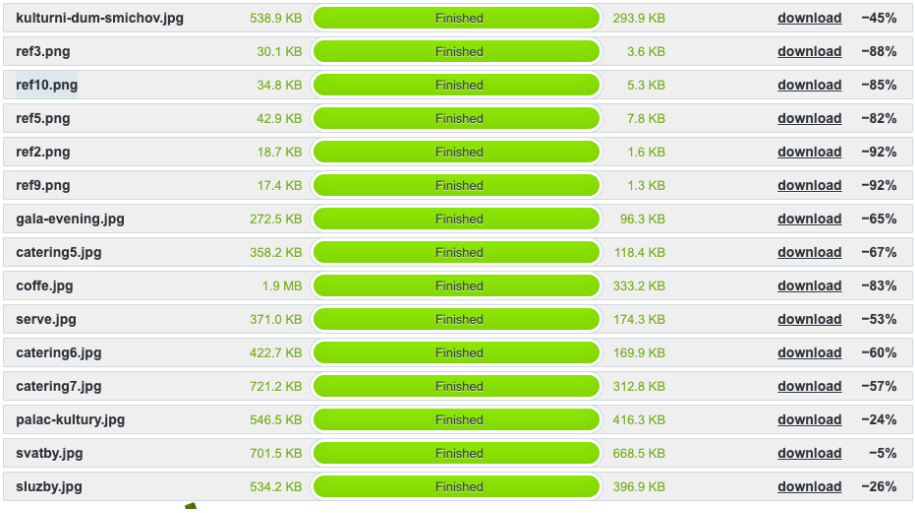
Obrázek 43 - Doporučená optimalizace obrázků pomocí nástroje Google PageSpeed Insights

V nástroji WebPageTest.org lze také odhalit obrázky, které lze zoptimalizovat. Po proběhnutí testu v záložce „Performance Review“ (tj. Výkonostní zpětná vazba).

Step_1	Keep-Alive 100%	GZip 66%	Compress Img 88%
1: jirikastanek.com - 1/	✓	⚠	
2: jirikastanek.com - all.css	✓	⚠	
3: jirikastanek.com - favicon.ico	✓	⚠	
4: jirikastanek.com - ref3.png	✓		⚠
5: jirikastanek.com - ref10.png	✓		⚠
6: jirikastanek.com - ref5.png	✓		⚠
7: jirikastanek.com - ref2.png	✓		⚠
8: blueimp.github.io...mp-gallery.min.css	✓	✓	
9: netdna.bootstrap... - font-awesome.css	✓	✓	
10: fonts.googleapis.com - css	✓	✓	
11: maxcdn.bootstra... - bootstrap.min.js	✓	✓	
12: jirikastanek.com - ref9.png	✓		⚠
13: ajax.googleapis.com - jquery.min.js	✓	✓	
14: blueimp.github....eimp-gallery.min.js	✓	✓	
15: www.google.com - embed	✓	✓	
16: jirikastanek.com - sluzby.jpg	✓		✓
17: jirikastanek.com - svatby.jpg	✓		✓
18: jirikastanek.com - gala-evening.jpg	✓		⚠
19: jirikastanek.com - catering5.jpg	✓		⚠
20: jirikastanek.com - coffe.jpg	✓		⚠
21: jirikastanek.com - serve.jpg	✓		⚠
22: fonts.gstatic.c...BA-UN7rgOUuhp.woff2	✓		
23: jirikastanek.com - koktejly.jpg	✓		✓
24: fonts.gstatic.c...MiZpBA-UFVZ0b.woff2	✓		
25: fonts.gstatic.c...BA-UN_r8OUuhp.woff2	✓		
26: jirikastanek.com - grill.jpg	✓		✓
27: fonts.gstatic.c...UN_r8OXOhpOqc.woff2	✓		
28: fonts.gstatic.c...UN7rgOXOhpOqc.woff2	✓		
29: jirikastanek.com - catering6.jpg	✓		⚠
30: jirikastanek.com - catering7.jpg	✓		⚠
31: jirikastanek.com - palac-kultury.jpg	✓		✓
32: jirikastanek.com...ni-dum-smichov.jpg	✓		⚠

Obrázek 44 - Doporučená optimalizace obrázků pomocí nástroje WebPageTest.org

Pro optimalizaci obrázků typu PNG či JPG bude využit nástroj TinyPNG, který je dostupný ve své základní verzi pro všechny. Tento nástroj je dostupný online přes webové rozhraní, kam se obrázky mohou nahrát všechny najednou. V základní verzi tento nástroj umožňuje nahrát až 20 obrázků najednou v maximální velikosti 5 MB pro každý obrázek.



kulturni-dum-smichov.jpg	538.9 KB	Finished	293.9 KB	download	−45%
ref3.png	30.1 KB	Finished	3.6 KB	download	−88%
ref10.png	34.8 KB	Finished	5.3 KB	download	−85%
ref5.png	42.9 KB	Finished	7.8 KB	download	−82%
ref2.png	18.7 KB	Finished	1.6 KB	download	−92%
ref9.png	17.4 KB	Finished	1.3 KB	download	−92%
gala-evening.jpg	272.5 KB	Finished	96.3 KB	download	−65%
catering5.jpg	358.2 KB	Finished	118.4 KB	download	−67%
coffe.jpg	1.9 MB	Finished	333.2 KB	download	−83%
serve.jpg	371.0 KB	Finished	174.3 KB	download	−53%
catering6.jpg	422.7 KB	Finished	169.9 KB	download	−60%
catering7.jpg	721.2 KB	Finished	312.8 KB	download	−57%
palac-kultury.jpg	546.5 KB	Finished	416.3 KB	download	−24%
svatby.jpg	701.5 KB	Finished	668.5 KB	download	−5%
sluzby.jpg	534.2 KB	Finished	396.9 KB	download	−26%

Obrázek 45 - Výsledek optimalizace obrázků pomocí nástroje TinyPNG

Po optimalizaci obrázků se zmenšila velikost obrázků celkem o 54 %, tedy o 4 MB. Největší optimalizace velikosti obrázků dosáhla zmenšení o 92 % z původní velikosti.

Další optimalizace se zaměřuje na načítání webového písma. Jedná se spíše o zacílení na uživatelský dojem, jelikož ve chvíli, kdy webové stránky stahují webový font, nějaký čas trvá, než se písmo načte. Nástroj Google PageSpeed Insights doporučuje skrze svoji příručku pro webové vývojáře použít do URL parametru *display=swap*. Pokud je webové písmo umístěno na vlastním hostingu, lze do stylu k písmu přidat CSS vlastnost *font-display: swap*.

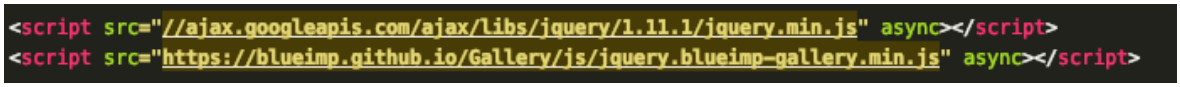
```
<link href='//fonts.googleapis.com/css?family=Open+Sans:400,700,300&display=swap' rel='stylesheet' type='text/css'>
```

Obrázek 46 - Přidání parametru display=swap

Dalším krokem optimalizace je zbavit webové stránky vložených prvků neboli iframů. Tyto prvky stránku taktéž brzdí ve vykreslení. Jedná se o prvky, které mohou být do stránky dočteny později. Nástroj, který bude použit, se také používá pro tzv. lazy loading neboli pomalé načítání obrázků, ale měl by fungovat i v případě použití iframů. Knihovna je JavaScriptová a prvky, které má načíst později, se inicializují pomocí přidání atributu *class="lazyload"* a *data-src="pozadované_url"*.

Obrázek 47 - Aplikace lazyload JavaScriptové knihovny

Dále došlo k přidání atributů defer či async k veškerému JavaScriptovému kódu, který je do stránky načítán a není potřebný pro její vykreslení. Tím byla minimalizována kritická cesta. Tedy nepotřebné soubory a jejich načtení je odloženo.



Obrázek 48 - Příklad použití atributu async i JavaScriptových knihoven

Jak lze vidět z obrázku číslo 47 nástroj WebPageTest.org navrhoval zapnout gzip pro určité soubory, jako jsou například CSS styly, html soubory, a jiné. Zapnutí gzip musíme provést na webovém hostingu. V tomto případě bude vložen následující řádek do souboru .htaccess, který se nachází v hlavním adresáři webového hostingu.

```
AddOutputFilterByType DEFLATE text/css text/html text/plain application/json text/xml appli
cation/javascript
```

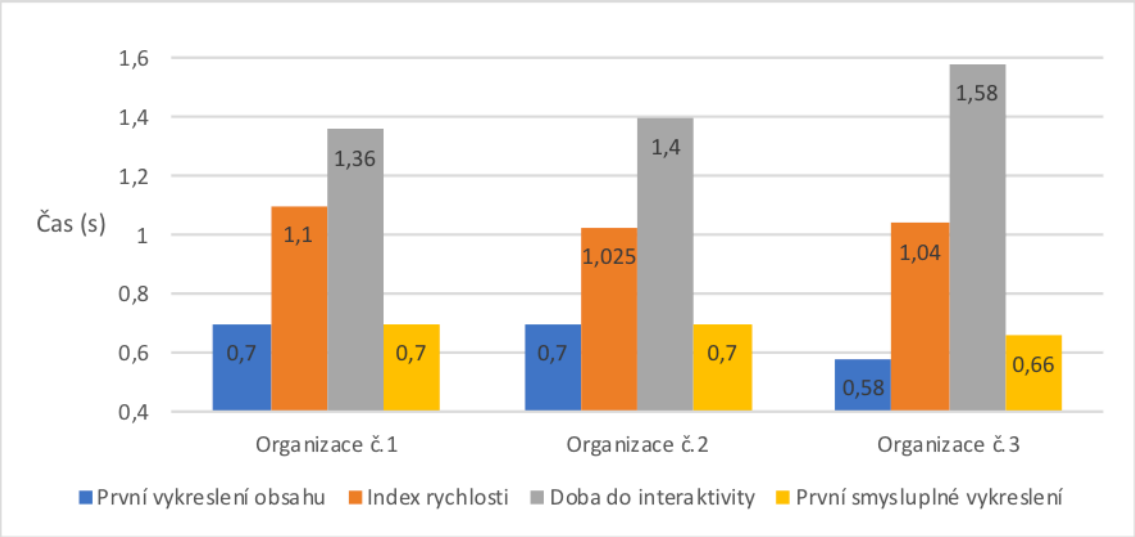
Obrázek 49 - Příklad zapnutí gzipu pro více typů souborů

5 Výsledky a diskuse

5.1 Porovnání organizací CSS stylů

Celkem byla provedena měření třech různých organizací CSS stylů. Počet prvků DOM stromu, které by mělo vliv na dané měření, zůstal stejný. Také jednotlivé styly nebyly po dobu měření nijak upravovány, došlo pouze k jejich různé organizaci v rámci webové stránky a pak její následnému načtení.

U každé organizace kaskádových stylů bylo provedeno měření, které bylo opakováno pětkrát. Z následných hodnot byl udělán aritmetický průměr, který byl následně porovnáván u naměřených metrik v rámci různých organizací.



Obrázek 50 - Vývoj vybraných metrik v jednotlivých organizacích CSS stylů

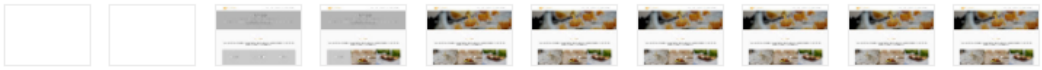
Na obrázku č.47 se jedná o srovnání organizace č.1 (tj. styly v jednom souboru), dále organizace CSS stylů č.2 (tj. styly jsou organizovány do několika menších celků), dále organizace CSS stylů č.3 (tj. styly jsou organizovány tzv. komponentově). Ze zobrazených dat lze vidět, že u metriky „První vykreslení obsahu“ došlo ke zlepšení až u posledního stylu organizace stylů – tedy komponentová organizace stylů v proběhlém měření dopadla nejlépe. Tento fakt lze zdůvodnit tím, že v okamžiku vykreslování se aplikují pouze ty CSS styly, které jsou potřeba k vykreslení nikoliv celý soubor CSS pravidel.

Naopak u metriky „Doba do interaktivity“, což může být chápáno jako doba, od které může uživatel plnohodnotně využít stránky, je vidět, že u poslední organizace oproti předešlým došlo ke zhoršení. Tedy organizace CSS stylů do jednoho souboru v měření

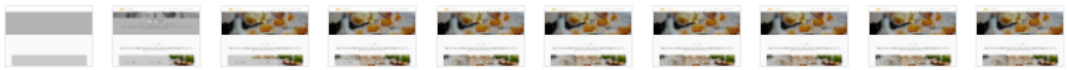
měla nejlepší hodnoty. Je to zároveň jediná metrika, u které je vidět zhoršení napříč měřeními. U ostatních metrik došlo naopak ke snižováním hodnot v řádech až desetin sekund, což u načítání webových stránek je velmi podstatná hodnota – tedy hlavně u větších webů. U menších webů je tato hodnota zanedbatelná.

Metrika „Index rychlosti“ byla nejmenší u organizace č.2 – tedy organizace orientována do menších celků a následně načtena v hlavičce stránky. Metrika „První smysluplné vykreslení“, které dává uživateli vizuální dojem, že se stránka načítá velmi rychle a nečeká na načtení obsahu, například který není vidět, došlo zlepšení – tedy snížení až u třetí organizace CSS stylů.

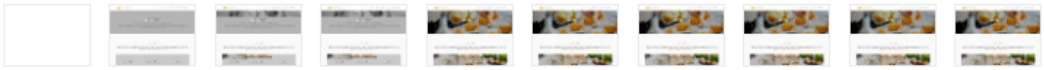
Při pozorování průběhu načtení webové stránky, které je vidět na obrázcích č. 51, 52 a 53, rozdělení hlavního souboru s kaskádovými styly do menších celků způsobilo rychlejší načtení a vykreslení prvního zobrazitelného obsahu.



Obrázek 51 - První způsob organizace kaskádových stylů



Obrázek 52 - Druhý způsob organizace kaskádových stylů



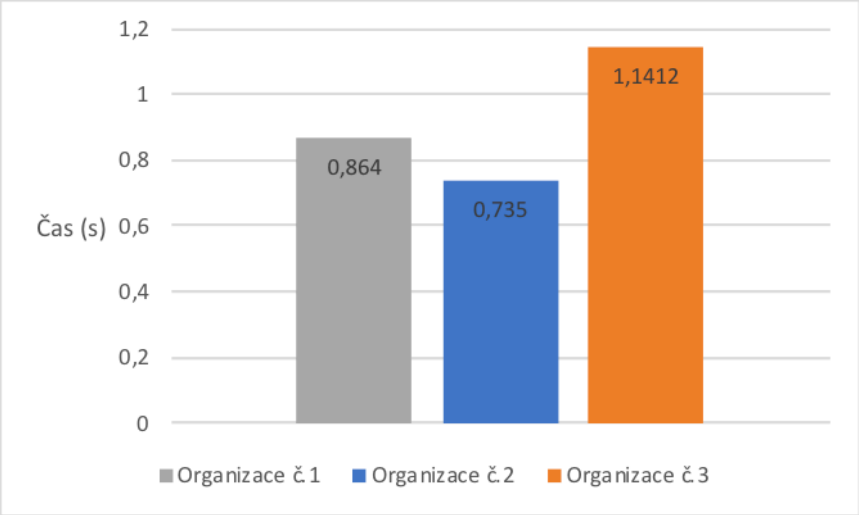
Obrázek 53 - Třetí způsob organizace CSS stylů

Při porovnání průběhu načtení druhého a třetího způsobu organizace kaskádových stylů, vizuální zobrazení postupného načítání stránek dopadlo lépe pro druhou nikoliv pro komponentově orientovanou. Tento vizuální test jasně dokresluje číselný výsledek předchozího měření.

Vykreslení	Organizace CSS stylů		
	č.1	č.2	č.3
Rendering	0,806	0,682	1,074
Layout	0,058	0,053	0,0672
Celkem	0,864	0,735	1,1412

Tabulka 23 - Srovnání vykreslení jednotlivých organizací CSS stylů

Samotné vykreslení vzorové webové stránky dopadlo nejlépe pro organizaci stylů č.2 – tedy několik menších souborů CSS, které jsou připojeny do hlavičky. Jako druhá dopadla organizace stylů v jednom souboru. Nejhůře v tomto měření dopadla komponentová organizace, kdy jednotlivé malé CSS soubory byly připojeny vždy před daným prvkem na stránce. Tuto skutečnost si lze vysvětlit tak, že prohlížeč vždy musel čekat s vykreslením stránky, než se daný malý soubor CSS stáhnul – tato skutečnost zastavila vykreslování. Proto komponentová organizace CSS stylů dosáhla ze všech tří organizací nejdelšího času vykreslení. Kdyby organizace byla přizpůsobena tak, že CSS kód by sice byl organizován podle komponent, ale při vykreslení by se vytvořil pouze jeden načítaný CSS soubor, dalo by se říci, že provedené měření by bylo nejkratší. Tento způsob dnes využívá mnoho JavaScriptových frameworků.



Obrázek 54 - Srovnání samotného vykreslení jednotlivých organizací CSS stylů

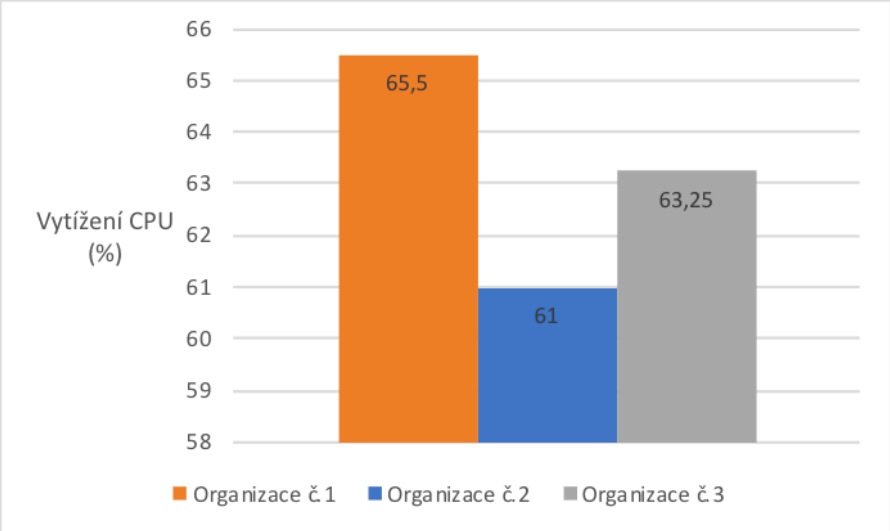
Měření vytížení počítače při vykreslení vzorových webových stránek byl zkoumán hlavní proces – tedy vlákno, které má na starosti vykreslení webové stránky. Následně pomocný proces GPU, který pomocí zapojení grafické karty zrychluje vykreslení.

Organizace stylů	Obsazená paměť [MB]	CPU [%]
č.1	466,2	65,5
č.2	428	61
č.3	447,1	63,25

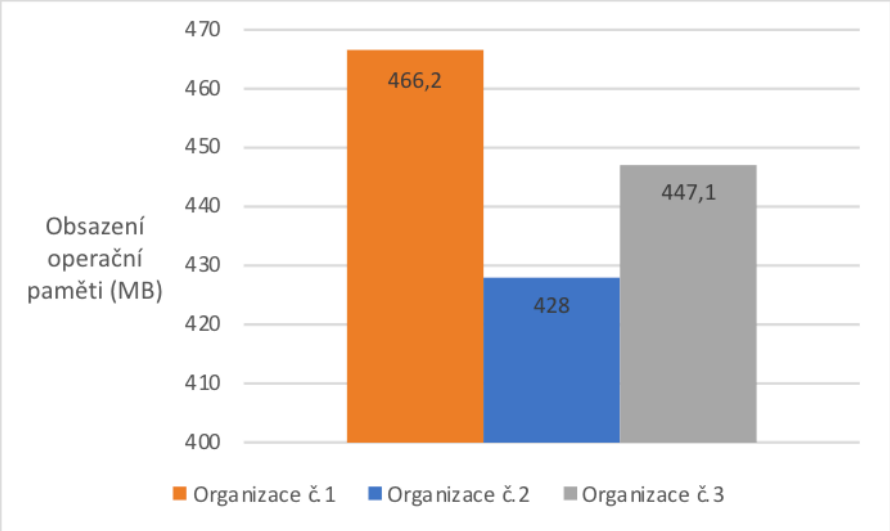
Tabulka 24 - Srovnání jednotlivých organizací CSS stylů a jejich vytížení klienta

Zatížení procesoru klienta dopadlo nejlépe pro organizaci stylů č.2. Jedná se o několik menších CSS souborů, které jsou připojené do hlavičky webu. Organizace stylů

tzv. komponentově (tj. č.3) skončila s hodnotu 63,25 % na místě druhém. CSS styly organizované do jednoho souboru (tj. č.1) vytěžují procesor nejvyšší hodnotou 65,5 %.



Obrázek 55 - Vytížení CPU v jednotkách procent



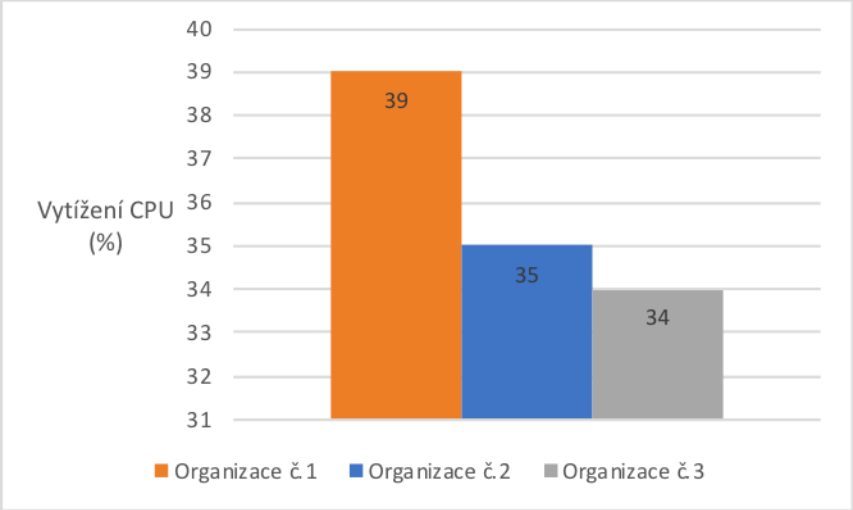
Obrázek 56 - Vytížení operační paměti [MB]

Při zkoumání spotřeby RAM paměti, tedy operační paměti, dopadla nejlépe organizace CSS stylů č.2, následně č.3 a poté až organizace stylů do jednoho velkého souboru.

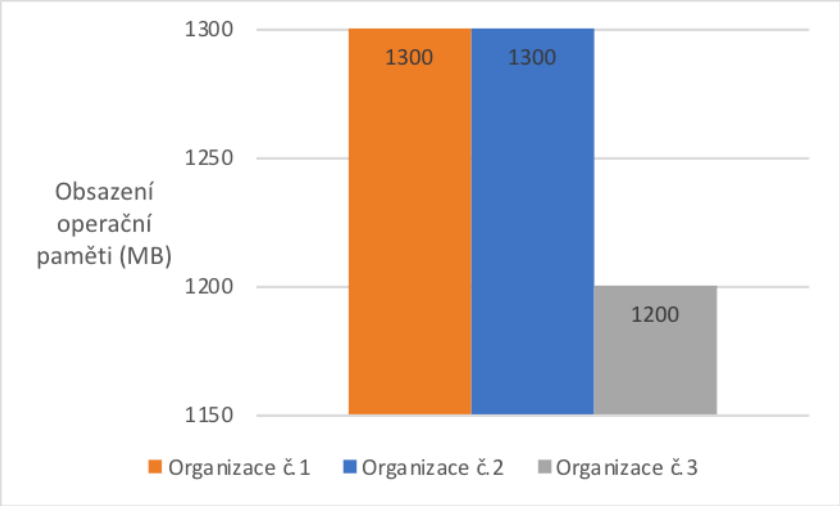
Organizace stylů	Obsazená paměť [MB]	CPU [%]
č.1	1300	39
č.2	1300	35
č.3	1200	34

Tabulka 25 - Srovnání jednotlivých organizací CSS stylů při zapnuté akceleraci GPU

Pomocný proces GPU obsadil na straně klienta nejméně paměti v organizaci stylů č.3 – tedy komponentově orientované. V rámci zatížení procesoru došlo ke stejnému výsledku.



Obrázek 57 - Vytížení CPU akcelerací pomocí GPU



Obrázek 58 - Vytížení operační paměti akcelerací pomocí GPU

5.2 Srovnání drahých CSS vlastností

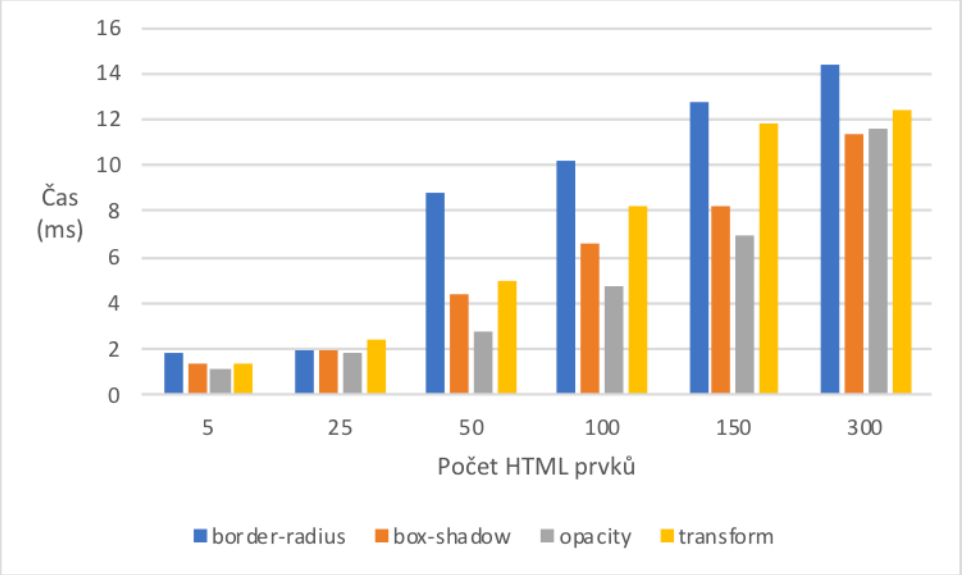
Na základě výsledků měření jednotlivých CSS vlastností a jejich srovnání s a bez použití dané vlastnosti lze u některých vlastností prokazatelně určit od jakého počtu aplikování daného pravidla dochází k prodlužování doby vykreslení zkoumaných HTML prvků.

Počet prvků	Rozdíl časů s a bez vlastnosti [ms]			
	border-radius	box-shadow	opacity	transform
5	1,8	1,4	1,2	1,4
25	2	2	1,8	2,4
50	8,8	4,4	2,8	5
100	10,2	6,6	4,8	8,2
150	12,8	8,2	7	11,8
300	14,4	11,4	11,6	12,4

Tabulka 26 - Naměřené časové rozdíly u jednotlivých CSS vlastností

U vlastnosti *border-radius* došlo k výraznému rozdílu času vykreslení s a bez použití dané vlastnosti od počtu 25 prvků, na které byla daná vlastnost aplikována pomocí CSS pravidla *.obdelnik*. Lze tedy u této vlastnosti prokazatelně říci, že by se měla používat v menší míře, a to při aplikování dané vlastnosti maximálně na počet 25 HTML prvků. Od již zmíněného počtu totiž dochází ke skokovému zvýšení rozdílu při použití či nepoužití dané vlastnosti.

U vlastnosti *box-shadow* došlo k více jak dvojnásobnému rozdílu u 50 HTML prvků na které se tato vlastnost pomocí CSS pravidla .obdelnik aplikovala. Lze tedy stejně tak jako u předchozí vlastnosti říci, že tuto vlastnost se v rámci času vykreslení vyplatí používat do 25 HTML prvků, na které se vlastnost aplikuje.



Obrázek 59 - Srovnání rozdílu časů s a bez použití jednotlivých vlastností

U vlastnosti *opacity* nedošlo k jednoznačnému určení, od jakého počtu HTML prvků, na které se aplikuje daná vlastnost, se nevyplatí používat vlastnost opacity. Jak lze vidět

z časových rozdílů (tj. s a bez vlastnosti) vývoj je pozvolný a u zvolených počtů HTML prvků nedošlo ke skokovému nárůstu času vykreslení. Pozvolný nárůst je samozřejmě ovlivněn velikostí DOM stromu, na který se daná vlastnost aplikuje.

U vlastnosti *transform* došlo k výraznému nárůstu rozdílu času vykreslení s a bez aplikování dané vlastnosti také od 25 HTML prvků. Tedy je doporučeno tuto vlastnost aplikovat maximálně na 25 HTML prvků.

5.3 Optimalizace stránek pro rychlejší načtení a vykreslení

Všechny kroky, které byly uskutečněny při optimalizaci, nemusí mít přímý vliv na snížení času načítání a následné vykreslení webových stránek.

Vybrané metriky	Před [s]	Po [s]
První vykreslení obsahu	0,7	0,5
Index rychlosti	1,1	0,8
Doba do interaktivity	1,36	1,9
První smysluplné vykreslení	0,7	0,5

Tabulka 27 - Srovnání časů metrik před a po optimalizaci

Po finálním přeměření byl zjištěn určitý posun, u metrik k menším, respektive kratším hodnotám. Metrika „První vykreslení obsahu“ se snížila o 0,2 sekundy. K tomuto posunu pomohlo zmenšení dat, které se musí stahovat při vykreslení stránky – jedná se především o datovém zmenšení obrázků a zapnutí komprimace Gzip. Metrika „Index rychlosti“ se zmenšila o 0,3 sekundy – tedy nedošlo k určitému zrychlení celkového vykreslení. Doba do interaktivity se naopak prodloužila o 0,54 s. Za tuto skutečnost může v první řadě načtení JavaScriptového kódu či knihoven třetích stran asynchronně, jeho načtení se odloží a web je tedy použitelný až později. V poslední řadě metrika „První smysluplné vykreslení“ se snížila o 0,2 sekundy. Tato metrika je stejně tak jako „První vykreslení obsahu“ důležitá pro uživatele, jelikož dochází u uživatele k pocitu, že se se stránkou něco děje rychleji. Uživatel vnímá určitý posun při vykreslení dříve, než kdyby se mu nejdříve zobrazila bílá stránka a po 0,6 sekundy naskočila celá stránka vykreslená.

	Před [s]	Po [s]
Rendering	0,806	0,8
Layout	0,058	0,049
Celkem	0,864	0,849

Tabulka 28 - Srovnání časů vykreslení před a po optimalizaci

Po optimalizaci došlo k zaměření rychlosti samotného vykreslení vzorové webové stránky. Ze srovnání před a po optimalizaci lze jednoznačně říci, že k určitému zlepšení času vykreslení došlo. Z hodnoty 0,864 sekund na 0,849, což je o 0,025 sekundy. U takto malého webu to není žádné velké zrychlení, ovšem představme si situaci, že takovýmto způsobem by se optimalizoval rozsáhlejší a větší web nebo informační systém v kterém by došlo k daleko větším rozdílům před a po optimalizaci webových stránek.

6 Závěr

V teoretické části práce byl vysvětlen pojem načítání webové stránky. Dále byl popsán základní princip vykreslovacího jádra prohlížeče. Byly popsány a rozebrány různé metodiky a organizace kaskádových stylů. V rámci měření rychlosti načítání, byly představeny různé metriky, které lze po měření vyhodnocovat. Následně byly popsány nástroje pro analýzu rychlosti načtení. V poslední řadě došlo k popsání faktorů, které mohou ovlivnit optimalizaci rychlosti načtení webové stránky.

K prokázání teoretických poznatků z první části práce, byla v praktické části vytvořena vzorová webová stránka, pomocí tří různých organizací CSS stylů. Na této webové stránce bylo na těchto třech organizacích provedeno několik zásadních měření. Došlo k různé organizaci v rámci vytvořené webové stránky a k jejímu načtení a změření časů načtení. Pro porovnání časů načtení a vykreslení byly zaměřeny tyto metriky: první vykreslení obsahu, index rychlosti, doba do interaktivity a první smysluplné vykreslení pomocí nástrojů PageSpeed Insights a Webpagetest.org. V rámci srovnání tří organizací kaskádových stylů v rychlosti načtení dosáhla nejkratšího času organizace č.2 (tj. rozdělení CSS pravidel do více menších souborů).

Dále také bylo zaměřeno vytížení na straně klienta, při načítání a vykreslení jednotlivých organizací kaskádových stylů. Měření bylo provedeno za pomoci správce procesů v prohlížeči Google Chrome. V tomto případě bylo sledováno vytížení operační paměti a procesoru. V zatížení procesoru dopadla nejlépe organizace CSS stylů č.2 (tj. rozdělení CSS pravidel do více menších souborů). Při zkoumání vytížení operační paměti byla nejmenší spotřeba RAM u organizace CSS stylů č.2.

Samotné vykreslení webové stránky bylo zaměřeno za pomocí prohlížeče Google Chrome, který poskytnul hodnoty Render a Layout, které dohromady dávají čas samotného vykreslení stránky v prohlížeči. V rámci výsledků samotného vykreslení webových stránek měla nejkratší čas vykreslení organizace č.2 (tj. rozdělení CSS pravidel do více menších souborů).

V rámci testování rychlosti načtení různých počtů použití drahých CSS vlastností, byla u vlastností border-radius, box-shadow a transform určena hranice počtu použití. Při použití většího počtu než 25 použití dané vlastnosti dochází k velkému nárůstu doby vykreslení. Pro vlastnost opacity nebyla jednoznačně prokázána určitá hranice, od které by docházelo k prodloužení doby vykreslení.

V rámci optimalizace vzorových webových stránek za pomoci zmenšení datové velikosti souborů, odložení stažení nepotřebných JavaScriptových knihoven třetích stran, zapnutí komprimace Gzip. Podle poznatků z teoretické části bylo dosaženo ke snížení času u metriky Index rychlosti, která vypovídá o celkové rychlosti načtení a vykreslení webu.

7 Seznam použitých zdrojů

[1] JANOVSKÝ, Dušan. HTTP protokol. *Jak psát web* [online]. [cit. 2019-08-21]. Dostupné z: https://www.jakpsatweb.cz/server/http-protokol.html

[2] JANOVSKÝ, Dušan. Na co stránka čeká. *Jak psát web* [online]. [cit. 2019-08-24]. Dostupné z: https://www.jakpsatweb.cz/clanky/na-co-stranka-ceka.html

[3] MOZILLA CONTRIBUTORS. Client-Server Overview. *MDN web docs* [online]. [cit. 2019-08-24]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Client-Server_overview

[4] BASQUES, Kayce. Chrome DevTools. *Tools for Web Developers* [online]. [cit. 2019-08-31]. Dostupné z: https://developers.google.com/web/tools/chrome-devtools/

[5] Zrychlení načítání stránek pomocí paralelních spojení. *Jak na webové stránky* [online]. [cit. 2019-08-31]. Dostupné z: http://timehosting.cz/paralelni-stahovani-prohlizecem-vice-hostname/

[6] ERLICH, Tomáš. Kompletní příručka pro zrychlení webu. *Interval.cz* [online]. 2015 [cit. 2019-08-31]. Dostupné z: https://www.interval.cz/clanky/kompletni-prirucka-pro-zrychleni-webu/

[7] SOUKUP, Petr. Jak se optimalizuje rychlost webů chytrým načítáním CSS. *Souki.cz* [online]. 2015 [cit. 2019-08-31]. Dostupné z: https://www.souki.cz/optimalizujeme-critical-asynchronni-css

[8] COYIER, Chris. Poll Results: Popularity of CSS Preprocessors. *CSS-TRICKS* [online]. 2016 [cit. 2019-09-10]. Dostupné z: https://css-tricks.com/poll-results-popularity-of-css-preprocessors/

[9] BRUNI, EZEQUIEL. 6 WAYS TO ORGANIZE YOUR CSS. *DEVELOPERdrive* [online]. [cit. 2019-09-10]. Dostupné z: https://www.developerdrive.com/6-ways-to-organize-your-css/

[10] BROWN, Inessa. Methods to Organize CSS. *CSS-TRICKS* [online]. 2017 [cit. 2019-09-10]. Dostupné z: https://css-tricks.com/methods-organize-css/

[11] MICHÁLEK, Martin. BEM: Pojmenovávací konvence pro třídy v CSS. *Vzhůru dolů* [online]. 2017 [cit. 2020-09-10]. Dostupné z: https://www.vzhurudolu.cz/prirucka/bem

[12] YANDEX CONTRIBUTORS. Key concepts. *BEM.info* [online]. [cit. 2019-09-10]. Dostupné z: https://en.bem.info/methodology/key-concepts/

[13] MICHÁLEK, Martin. BEM: Pojmenovávací konvence pro třídy v CSS. *Zdroják.cz* [online]. 2017 [cit. 2019-09-15]. Dostupné z: https://www.zdrojak.cz/clanky/bem-pojmenovavaci-konvence-tridy-css/

[14] ARSENAULT, Cody. OOCSS - The Future of Writing CSS. *KeyCDN* [online]. 2019 [cit. 2019-09-15]. Dostupné z: https://www.keycdn.com/blog/oocss

[15] MENDONCA, Claudio. How to Organize Your CSS with a Modular Architecture (OOCSS, BEM, SMACSS). *Snipcart* [online]. 2018 [cit. 2019-09-15]. Dostupné z: https://snipcart.com/blog/organize-css-modular-architecture

[16] MICHÁLEK, Martin. Utility CSS: K čemu jsou dobré systémy jednoúčelových tříd? *Vzhůru dolů* [online]. 2019 [cit. 2019-09-15]. Dostupné z: https://www.vzhurudolu.cz/prirucka/css-utility
[17] MICHÁLEK, Martin. Utility třídy a komponenty v CSS: Jak najít rovnováhu? *Vzhůru dolů* [online]. 2019 [cit. 2019-09-15]. Dostupné z: https://www.vzhurudolu.cz/prirucka/css-utility-komponenty
[18] CSS RYCHLE A EFEKTIVNĚ. *CSS3 Tipy a Triky* [online]. 2016 [cit. 2019-09-15]. Dostupné z: http://css.chobits.ch/css-rychle-a-efektivne/
[19] MICHÁLEK, Martin. Zásady psaní respektujícího CSS. *Vzhůru dolů* [online]. 2018 [cit. 2019-09-15]. Dostupné z: https://www.vzhurudolu.cz/prirucka/rcss-zasady
[20] BIEN, Jan. O rychlosti webu, asynchronním načítání, kritických CSS. *Webmistr* [online]. 2015 [cit. 2019-09-15]. Dostupné z: https://www.webmistr.wtf/rychlost-webu/
[21] EVERTS, Tammy a Tim KADLEC. WPO stats. *Wpostats* [online]. 2018 [cit. 2019-09-20]. Dostupné z: https://wpostats.com/
[22] MICHÁLEK, Martin. CSS: Optimalizace datové velikosti. *Vzhůru dolů* [online]. 2019 [cit. 2019-09-20]. Dostupné z: https://www.vzhurudolu.cz/prirucka/css-optimalizace
[23] HEJČ, Roman. Jak vyřešit rychlost načítání webu. *Shean* [online]. 2019 [cit. 2019-09-20]. Dostupné z: https://www.shean.cz/clanky/detail/jak-vyresit-rychlost-nacitani-webu.htm
[24] MICHÁLEK, Martin. Metriky rychlosti webů: Průvodce s detailním vysvětlením. *Vzhůru dolů* [online]. 2019 [cit. 2019-09-20]. Dostupné z: https://www.vzhurudolu.cz/prirucka/metriky-rychlosti
[25] W3C CONTRIBUTORS. OS Platform Statistics. *W3schools* [online]. 2003 [cit. 2019-09-20]. Dostupné z: https://www.w3schools.com/browsers/browsers_os.asp
[26] PAGESPEED INSIGHTS CONTRIBUTORS. About PageSpeed Insights. *PageSpeed Insights* [online]. 2019 [cit. 2019-09-24]. Dostupné z: https://developers.google.com/speed/docs/insights/v5/about
[27] MICHÁLEK, Martin. Co je „Doba do načtení prvního bajtu“ (aneb „Time To First Byte“ aneb TTFB)? *Vzhůru dolů* [online]. 2019 [cit. 2019-09-20]. Dostupné z: https://www.vzhurudolu.cz/prirucka/ttfb
[28] MICHÁLEK, Martin. Událost DOM Content Loaded (DCL). *Vzhůru dolů* [online]. 2019 [cit. 2019-10-10]. Dostupné z: https://www.vzhurudolu.cz/prirucka/udalost-dcl
[29] MICHÁLEK, Martin. Událost „První vykreslení“ (First Paint, FP). *Vzhůru dolů* [online]. 2019 [cit. 2019-10-15]. Dostupné z: https://www.vzhurudolu.cz/prirucka/metrika-fp
[30] MICHÁLEK, Martin. Metrika „První vykreslení obsahu“ (First Contentful Paint, FCP). *Vzhůru dolů* [online]. 2019 [cit. 2019-10-15]. Dostupné z: https://www.vzhurudolu.cz/prirucka/metrika-fcp

[31] TOOLS FOR WEB DEVELOPERS CONTRIBUTOR. First Contentful Paint. *Tools for Web Developers* [online]. 2019 [cit. 2019-10-15]. Dostupné z: https://developers.google.com/web/tools/lighthouse/audits/first-contentful-paint
[32] WALTON, Philip. First Input Delay (FID). *Web.dev* [online]. 2020 [cit. 2019-10-15]. Dostupné z: https://web.dev/fid/
[33] MICHÁLEK, Martin. Metrika „První smysluplné vykreslení“ (First Meaningful Paint, FMP). *Vzhůru dolů* [online]. 2019 [cit. 2019-10-15]. Dostupné z: https://www.vzhurudolu.cz/prirucka/metrika-fmp
[34] TOOLS FOR WEB DEVELOPERS CONTRIBUTOR. Time to Interactive. *Tools for Web Developers* [online]. 2019 [cit. 2019-10-20]. Dostupné z: https://developers.google.com/web/tools/lighthouse/audits/time-to-interactive
[35] AOL CONTRIBUTOR. Speed Index. *WebPagetest Documentation* [online]. [cit. 2019-10-20]. Dostupné z: https://sites.google.com/a/webpagetest.org/docs/using-webpagetest/metrics/speed-index
[36] MICHÁLEK, Martin. Metrika „Index rychlosti“ (Speed Index, SI). *Vzhůru dolů* [online]. 2019 [cit. 2019-10-20]. Dostupné z: https://www.vzhurudolu.cz/prirucka/speedindex
[37] WALTON, Philip. User-centric performance metrics. *Web.dev* [online]. 2019 [cit. 2019-10-20]. Dostupné z: https://web.dev/user-centric-performance-metrics/
[38] MICHÁLEK, Martin. PageSpeed Insights: Kompletní průvodce testem rychlosti webu. *Vzhůru dolů* [online]. 2019 [cit. 2019-10-20]. Dostupné z: https://www.vzhurudolu.cz/prirucka/pagespeed-insights
[39] MICHÁLEK, Martin. Vzhůru do (responzivního) webdesignu. Verze 1.1. Praha: vlastním nákladem autora, 2017. ISBN 978-80-88253-00-6.
[40] PAGESPEED INSIGHTS. Optimalizační stránka. In: *PageSpeed Insights* [online]. [cit. 2019-10-20]. Dostupné z: https://developers.google.com/speed/pagespeed/insights/?hl=cs&url=http%3A%2F%2Fpef.czu.cz%2Fcz&tab=desktop
[41] MICHÁLEK, Martin. PageSpeed Insights: Podrobný průvodce nástrojem pro měření rychlosti. *Zdroják.cz* [online]. 2018 [cit. 2019-10-20]. Dostupné z: https://www.zdrojak.cz/clanky/pagespeed-insights-podrobny-pruvodce-nastrojem-pro-mereni-rychlosti/
[42] AOL CONTRIBUTOR. WebPagetest Documentation. *WebPagetest* [online]. 2019 [cit. 2019-10-20]. Dostupné z: https://sites.google.com/a/webpagetest.org/docs/
[43] AOL CONTRIBUTOR. Test a website's performance. *Webpagetest.org* [online]. [cit. 2019-10-20]. Dostupné z: https://www.webpagetest.org/
[44] Latest Performance Report for. *GTmetrix* [online]. 2019 [cit. 2019-10-20]. Dostupné z: https://gtmetrix.com/reports/pef.czu.cz/nzAtJecQ
[45] IRISH, Paul a Tali GARSIEL. How Browsers Work: Behind the scenes of modern web browsers. *Html5rocks* [online]. 2011 [cit. 2019-10-22]. Dostupné z: https://www.html5rocks.com/en/tutorials/internals/howbrowserswork/

[46] JAHODA, Bohumil. Jak funguje vykreslování stránky. *Jecas.cz* [online]. 2014 [cit. 2019-10-22]. Dostupné z: https://jecas.cz/vykreslovani
[47] W3C CONTRIBUTORS. Browser Statistics. *W3schools* [online]. 2002 [cit. 2019-10-22]. Dostupné z: https://www.w3schools.com/browsers/
[48] GRIGORIK, Ilya. Render-tree Construction, Layout, and Paint. *Web Fundamentals* [online]. 2019 [cit. 2019-10-22]. Dostupné z: https://developers.google.com/web/fundamentals/performance/critical-rendering-path/render-tree-construction
[49] BUCKLER, Craig. 10 Ways to Minimize Reflows and Improve Performance. *Sitepoint* [online]. 2015 [cit. 2019-10-22]. Dostupné z: https://www.sitepoint.com/10-ways-minimize-reflows-improve-performance/
[50] BROWN, Tiffany B. CSS Master. Austrálie: Sitepoint, 2015. ISBN 978-0-9941826-2-3
[51] BUDD, Andy, Cameron MOLL a Simon COLLISON. CSS mastery: advanced web standards solutions. 2nd. ed. New York: Distributed to the Book trade in the United States by Springer-Verlag New York, 2009. ISBN 978-1-4302-2397-9.
[52] FRAIN, Ben. CSS performance revisited: selectors, bloat and expensive styles. *Benfrain* [online]. 2014 [cit. 2019-10-24]. Dostupné z: https://benfrain.com/css-performance-revisited-selectors-bloat-expensive-styles/
[53] GERARD, Charlie. Things nobody ever taught me about CSS. *Medium* [online]. 2019 [cit. 2019-10-24]. Dostupné z: https://medium.com/@devdevcharlie/things-nobody-ever-taught-me-about-css-5d16be8d5d0e
[54] MCANLIS, Colt. CSS Paint Times and Page Render Weight. *Html5rocks* [online]. 2013 [cit. 2019-10-24]. Dostupné z: https://www.html5rocks.com/en/tutorials/speed/css-paint-times/
[55] JANOVSKÝ, Dušan. CSS vlastnosti. *Jak psát web* [online]. 2019 [cit. 2019-10-24]. Dostupné z: https://www.jakpsatweb.cz/css/vlastnosti.html
[56] SOUDERS, Steve. High performance web sites: essential knowledge for frontend engineers. Farnham: O'Reilly, c2007. ISBN 978-0596529307.
[57] WAGNER, Jeremy L. a Ethan MARCOTTE. Web performance in action: building fast web pages. Shelter Island, NY: Manning Publications Co., [2017]. ISBN 978-1617293771.
[58] YAHOO CONTRIBUTORS. Best Practices for Speeding Up Your Web Site. *Yahoo Developer* [online]. 2019 [cit. 2019-10-25]. Dostupné z: https://developer.yahoo.com/performance/rules.html
[59] PODJARNY, Guy. Responsive & Fast: Implementing High-Performance Responsive Design. Sebastopol, CA: O'Reilly Media, c2014. ISBN 978-1491911617.
[60] JAHODA, Bohumil. Data URI. *Jecas* [online]. 2015 [cit. 2019-10-25]. Dostupné z: https://jecas.cz/data-uri
[61] W3TECHS CONTRIBUTORS. Usage statistics of HTTP/2 for websites. *W3techs* [online]. 2019 [cit. 2019-10-25]. Dostupné z: https://w3techs.com/technologies/details/ce-http2

[62] JAHODA, Bohumil. K čemu slouží CDN? *Jecas* [online]. 2016 [cit. 2019-10-25]. Dostupné z: https://jecas.cz/cdn
[63] W3TECHS CONTRIBUTORS. Usage statistics of Gzip Compression for websites. *W3techs* [online]. 2019 [cit. 2019-10-25]. Dostupné z: https://w3techs.com/technologies/details/ce-gzipcompression
[64] GRIGORIK, Ilya. Optimizing Encoding and Transfer Size of Text-Based Assets. *Web Fundamentals* [online]. 2019 [cit. 2019-10-26]. Dostupné z: https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/optimize-encoding-and-transfer#text-compression-with-gzip
[65] GRIGORIK, Ilya. Image Optimization. *Web Fundamentals* [online]. 2019 [cit. 2019-10-26]. Dostupné z: https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/image-optimization
[66] MICHÁLEK, Martin. Lazy loading obrázků a iframe: Kompletní průvodce implementací (včetně nativní podpory v prohlížečích). *Vzhůru dolů* [online]. 2019 [cit. 2019-10-26]. Dostupné z: https://www.vzhurudolu.cz/prirucka/lazy-loading-obrazku
[67] OSMANI, Addy. Web Performance Optimization with webpack. *Web Fundamentals* [online]. 2019 [cit. 2019-10-26]. Dostupné z: https://developers.google.com/web/fundamentals/performance/webpack/
[68] JANČA, Marek. Webpack – moderní Web Development. *Ackee* [online]. 2019 [cit. 2019-10-27]. Dostupné z: https://www.ackee.cz/blog/moderni-web-development-webpack/