UNVIERZITA PALACKÉHO V OLOMOUCI

PEDAGOGICKÁ FAKULTA

Katedra technické a informační výchovy

Bakalářská práce

Daniel Kuchař

Tvorba webových stránek pro různá zobrazovací zařízení

Olomouc 2021

vedoucí práce: Mgr. Tomáš Dragon

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně a uvedl jsem v ní veškerou literaturu a ostatní informační zdroje, které jsem použil.

V Olomouci dne 20. dubna 2021 vlastnoruční podpis ……

Poděkování

Chtěl bych poděkovat Mgr. Tomáši Dragonovi za pomoc a rady, které mi poskytl při tvorbě této práce a vstřícnost při konzultacích. Velké díky také patří mé rodině a přátelům za neustálou podporu.

Daniel Kuchař

Obsah

Úvod

Bakalářská práce s názvem *Tvorba webových stránek pro různá zobrazovací zařízení* je zaměřena na problematiku responzivity při tvorbě webových stránek.

Hlavním cílem práce je prakticky implementovat moderní webové technologie a postupy za účelem aktualizace vybrané webové stránky, která pak bude splňovat moderní standardy v oblasti responzivity. Dílčím cílem práce je představit moderní webové technologie a přístupy při tvorbě responzivního webdesignu.

Jelikož se webdesign jako disciplína v posledních letech neustále rozvíjí, je třeba specifikovat, že tato práce se zabývá především tzv. front-endem. Jedná se o klientskou stranu webdesignu, která se zabývá výstupním vzhledem stránek, a tudíž i aplikací procesů potřebných pro správnou přípravu stránek pro všechny druhy zobrazovacích zařízení. Tato příprava nespočívá pouze v grafických úpravách, díky kterým se stránky vizuálně mění v závislosti na velikosti zobrazovacího zařízení, ale také přispívá k tomu, aby byla stránka lépe dohledatelná a splňovala požadavky algoritmů internetových vyhledávačů a byla přístupná i pro návštěvníky se speciálními potřebami.

Práce je strukturována na dvě části – teoretickou a praktickou. Teoretická část se skládá ze čtyř kapitol. Jsou zde postupně vysvětlovány jednotlivé pojmy, aktuální moderní technologie, které jsou využitelné při tvorbě responzivních webových stránek, nebo mají za úkol jejich tvorbu ulehčovat, a také různé typy přístupů k tvorbě webových stránek. Praktická část práce je zaměřena na implementaci vybraných moderních webových technologií vhodných pro tvorbu responzivních webových stránek do reálného projektu, tj. aktualizace stávajícího kódu webové stránky podle nejnovějších standardů a zároveň také celková úprava vizuální části webu.

1 Úvod do problematiky moderních webových technologií

V souvislosti s dílčím cílem práce je nutné se nejprve zaměřit na základní pojmy, které se budou napříč touto prací často vyskytovat a je tudíž důležité znát jejich význam a koncept použitelnosti v programátorské praxi.

Webdesign

I když se jedná o často používaný termín, není snadné najít uspokojivou a jednoznačnou definici slova webdesign, zejména díky rychlému rozvoji v posledních letech. Někdy se tento pojem používá v kontextu tvorby celé webové stránky již od počáteční myšlenky. Po webdesignerovi je v tomto případě žádáno například blokové navržení stránky neboli wireframe[1], návrh a tvorba databáze nebo i marketingové prvky (ManagementMania, 2020).

Pro účely této práce se však spokojíme s jednodušší definicí a budeme se zabývat hlavně částmi webdesignu, které se týkají optimalizace pro různá zobrazovací zařízení, a o jiných aspektech se zmíníme jen okrajově.

Back-end

Proces tvorby webové stránky je zpravidla rozdělen na front-end a back-end. Back-end je svým způsobem zákulisí webové stránky, kde se ovládá veškeré dění na straně serveru. Hlavní náplní práce back-end programátora je objektově orientované programování, pomocí kterého řeší problémy jako například:

- Tvorba databáze a následné provázání se stránkou;
- Řešení ukládání, zpracovávání a šifrování dat;
- Testování funkčnosti jednotlivých funkčních částí stránky;
- Zajištění, aby stránkám nehrozil kolaps při vyšší návštěvnosti;
- Užívání programovacích jazyků jako jsou .NET, C++ nebo MySQL;
- Dodatečné doplňování funkčních prvků a aktualizace (Castiglione, 2019).

[1] Jedná se o velmi jednoduché grafické rozložení, složené zejména ze základních obrazců reprezentujících jednotlivé elementy nebo oddíly

Front-end

Front-end se na rozdíl od back-endu zabývá tím, jak stránky vypadají a jak k nim přistupují uživatelé. To je také hlavním bodem zájmu této práce.

Na front-end straně řešíme a implementujeme vše, co se týká vzhledu a uživatelské interakce. I na front-end scéně se neobejdeme bez programování, avšak oproti back-endu nepotřebujeme pokročilé znalosti objektově orientovaného programování a většinou si vystačíme i se základními programovacími a skriptovacími jazyky jako jsou HTML, CSS a JavaScript. Hlavní pracovní náplní front-end kodéra je například:

- Tvorba a úprava uživatelského rozhraní;
- Kódování samotné stránky, sloužící jako interakce mezi uživatelem a serverem;
- Zaručení správné funkčnosti stránek po vzhledové stránce – responzivita, přístupnost a intuitivnost (Castiglione, 2019).

Framework

Frameworky jsou vytvořené sady, které mají za účel kodérovi usnadnit práci v konkrétním odvětví a jazyce. Může se jednat například o usnadnění zápisu, kdy kodér může použít například již existující sadu tříd, které mají přednastavenou definici v CSS a ušetřit čas tím, že nebude psát stále stejná pravidla znovu a znovu. V našem případě se budeme zajímat hlavně o responzivní frameworky, které usnadňují tvorbu responzivních stránek a jsou postaveny zejména na HTML a CSS, jako například Twitter Bootstrap[2] nebo ZURB Foundation[3] (Twitter, 2020, ZURB, 2020).

V této práci k nim budeme referovat jako ke frameworkům, přestože některé zdroje uvádějí, že se nejedná o frameworky ale toolkity[4] nebo knihovny. Bohužel termíny jako Framework, knihovna a toolkit si jsou velmi blízké a s odstupem času se jejich definice více a více specifikují. Takže i v publikacích bývají často zaměňovány (Wozniewicz, 2019).

[2] Dostupné z: www.getbootstrap.com
[3] Dostupné z: www.get.foundation
[4] Toolkit označuje softwarovou sadu nástrojů

V dnešní době jsou za front-end frameworky zejména chápány takové, které pracují spíše s jazykem JavaScript a jejich hlavním smyslem je vytvoření struktury stránky, do které dále vkládáme svůj kód. Mezi tyto frameworky patří například Angular nebo React (Pekarsky, 2020).

Mimo front-end samozřejmě máme i back-end frameworky, které programátorovi pomáhají například pro řešení bezproblémové komunikace s databází nebo ochrany kódu před útoky. Mezi tyto frameworky patří například Ruby on Rails nebo Spring Boot (Castiglione, 2019).

Jak jsem již dříve zmínil, často se setkáváme i s pojmem knihovna, který někdy bývá zaměňován se slovem framework. Princip knihoven je velmi podobný, avšak jedná se o odlehčené verze, které obsahují pouze procedury a funkce, ze kterých si sami vybíráme a voláme je, kdy je podle nás potřeba. Frameworky jsou však o něco komplikovanější a rozsáhlejší. Místo abychom si z frameworku vybíráme pouze to, co potřebujeme, framework nám většinou naservíruje již přednastavenou konstrukci, do které dále doplňujeme náš kód podle jeho pravidel (Wozniewicz, 2019).

1.1 Layout

Layout[5] nám udává obecné blokové uspořádání stránky. Našim cílem je vždy vytvářet layout, který se přizpůsobuje obrazovce a je kompletně viditelný.

Mimo responzivních funkcí je také hlavním aspektem layoutu cílová skupina a záměr stránky. Tím myslíme, že webová stránka sloužící jako portfolio bude používat jiný layout než například e-shop. Touto stránkou layoutu se však v této práci zabírat nebudeme a rozebereme si hlavně problematiku responzivity (Raducan, 2018).

1.1.1 Statický – Fixed layout

Jedná se o layout s pevně danou šířkou (většinou v jednotkách pixelů). Všechen obsah uvnitř fixního layoutu je tedy omezován touto šířkou a v něm zahrnuté elementy se od této šířky budou odvozovat. Pokud tedy nastavíme šířku obsahu na 1000 pixelů, na obrazovkách s menší

[5] Jedná se o ekvivalent výrazu rozložení nebo uspořádání.

šířkou uvidíme jen tolik, kolik nám displej dovoluje, nebo budeme muset pro prohlédnutí celého obsahu posouvat stránkou do stran (UX Alpaca, 2016).

1.1.2 Dynamický – Fluid layout

Jak již název napovídá, v případě fluid layoutu mluvíme o přizpůsobivé variantě. Šířka dynamického rozložení je dána většinou v procentech, ale můžeme použít i univerzální jednotky jako například ‚em' nebo ‚rem'. Pokud tedy nastavíme šířku na 80 %, bude na každém zařízení zabírat 80% celkové šířky. Čistý fluid layout nepoužívá žádné body zlomu a za všech podmínek se řídí přepočtenými hodnotami (UX Alpaca, 2016).

1.2 Responzivita

Tento pojem je pro nás stěžejní, jelikož je v souladu s hlavním cílem této práce. Napříč celou prací se tento pojem bude vyskytovat často a v různých scénářích.

Responzivita je v dnešní době jedním z nejdůležitějších aspektů webových stránek. Díky změnám ve vyhledávacích algoritmech online vyhledávačů má stránka, která je přehledná a dobře čitelná na všech zařízeních, mnohem větší šanci na výskyt ve vyhledávání než zastaralé stránky. Zde se dostáváme do stádia, kdy responzivita není pouze záležitostí intuitivnosti a vzhledu stránky, ale také funkčnosti, marketingu a přístupnosti pro různé návštěvníky (Xia, 2017).

Dosažení správné responzivity na stránkách se dá docílit několika způsoby. Avšak pro všechny platí některá základní pravidla. Jedním z těchto pravidel se stalo co největší využívání globálních velikostí místo pevných. Tím jsou myšleny například jednotky %, em, rem, vw a vh. Tyto jednotky odvíjejí svou velikost podle zbytku stránky nebo rozlišení prohlížeče, na rozdíl od pixelů, které svou velikost drží (Bracey, 2015).

Hlavní pravidlo při tvorbě responzivity moderních stránek je založeno na vytváření tzv. breakpointů, což jsou body zlomu v různých rozlišeních, ve kterých se aplikují nová pravidla pro vzhled. K vytváření těchto bodů používáme jazyk CSS, ve kterém specifikujeme pravidla pomocí Media query(viz *kapitola 2.2.1*). Tato funkce nám umožňuje zaměření na konkrétní typ zařízení i rozlišení, čímž vytváříme dříve zmiňované breakpointy. Pro orientaci si uvedeme některé základní záchytné body v podobě šířek zařízení, které jsou běžně využívány pro vytváření zlomů:

- 320–480 pixelů pro mobilní zařízení,
- 481–768 pixelů pro tablety a iPady,
- 769–1024 pixelů pro laptopy a malé obrazovky,
- 1025–1200 pixelů pro desktopová zařízení a velké obrazovky,
- 1201 pixelů a více pro extra velké obrazovky a televize (Eygi, 2020).

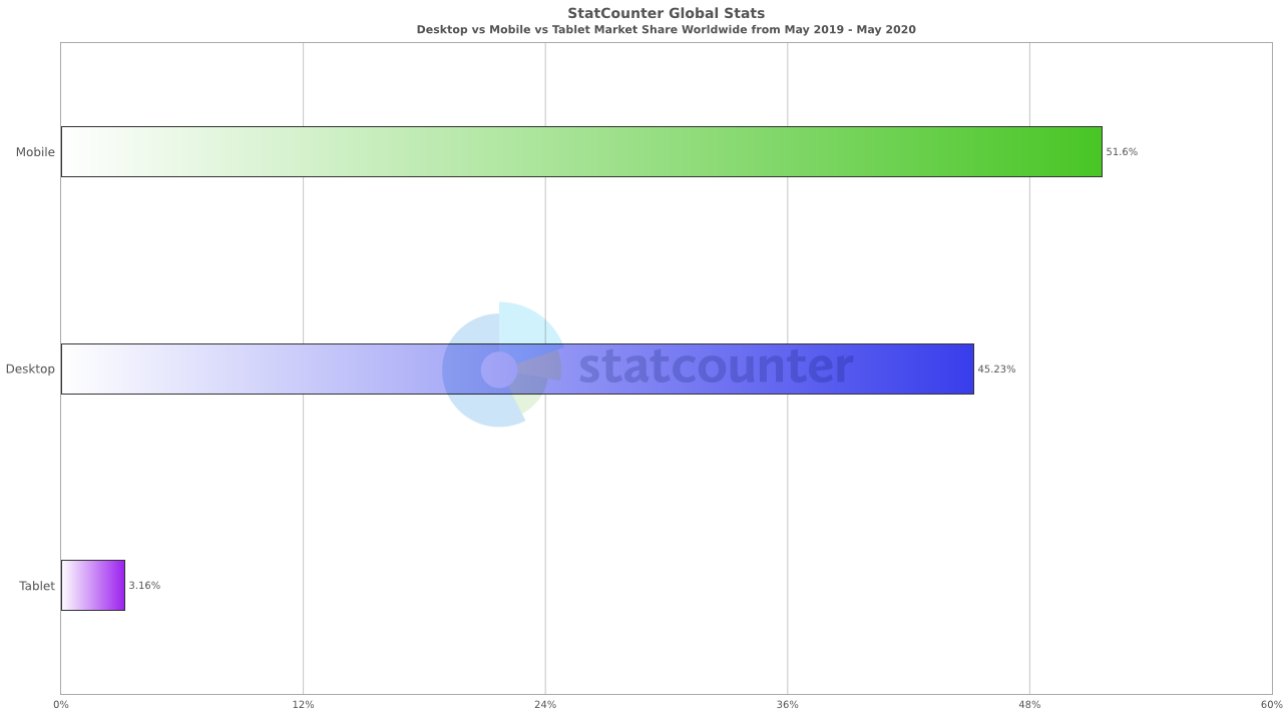
Pokud však již nějaké stránky máme a chceme ověřit její responzivitu, máme na výběr několik možností. Jednou z nich je online kontrola responzivity, dostupná například na stránce *Test použitelnosti v mobilech – Google Search Console*[6]. Na pokročilejších kontrolách jako je tato, dostaneme i výpis chyb a problémů, které je třeba napravit pro bezchybnou responzivitu.

Stejně tak můžeme najít na internetu mnoho emulátorů mobilních zobrazení jako například *Responsive Website Design Tester*[7], kde si můžeme naši stránku zobrazit a zkoušet interakce v různých rozlišeních a na konkrétních zařízeních.

V mnoha případech si však vystačíme i s internetovým prohlížečem jako například Chrome nebo Firefox, které v sobě mají zabudovanou funkci emulace jiných zařízení. Jinak také můžeme zkoušet různá rozlišení zmenšováním a zvětšováním okna prohlížeče. Není však dobré se vždy spoléhat pouze na tyto emulace a je vhodné si stránky přímo kontrolovat i na všech možných zařízeních, které máme po ruce dostupné (LePage, Andrew, 2020).

Se záměrem zakomponování responzivity již jako základní součásti webových stránek se začaly objevovat nové přístupy k navrhování stránek i webových aplikací. Stále známějším se stává termín Mobile first design, reprezentující metodu tvorby webů/webových aplikací tak, že jako první navrhujeme vzhled a rozpoložení pro mobilní zařízení, a dále jej postupně rozšiřujeme na větší obrazovky. Metoda mobile first designu se stává stále populárnější zejména díky tomu, že v návštěvnosti převažují mobilní zařízení desktopová (viz *Obr. 1*) (Xia, 2017).

[6] Dostupné z: www.search.google.com/test/mobile-friendly
[7] Dostupné z: www.responsivedesignchecker.com



Obr. 1: Graf návštěvnosti webových stránek dle používaných zařízení za poslední rok

(Zdroj: Statcounter, 2020)

V další kapitole si představíme vybrané webové technologie, které jsou v současnosti dostupné pro tvorbu webových stránek.

2 Moderní webové technologie

V návaznosti na dílčí cíl práce si v této kapitole si představíme moderní webové technologie, které jsou běžně používány při tvorbě webových stránek pro různá zobrazovací zařízení. Jedná se zejména o programovací jazyky, knihovny a jejich konkrétní části.

2.1 HTML

Jedná se o značkovací jazyk, který je stavebním kamenem každé webové stránky. Již při psaní základní HTML struktury je vhodné, aby kodér myslel na responzivitu jednotlivých elementů. Správně naplánovaná struktura pomáhá předejít mnoha problémům ohledně správného zobrazení na různých zařízeních.

Od roku 2014 je standardně používána HTML verze 5, která předchozí verzi obohatila o mnoho nových elementů a atributů. V jazyce HTML nejen určujeme, jaký element je text, video, či obrázek, ale také každému elementu můžeme přiřadit tzv. atributy. Každý element má své specifické atributy pro různá využití. Existují však i atributy, které můžeme přidělit každému elementu. Jedná se hlavně o atributy tříd (class) a identifikátorů (ID), bez kterých se dnes takřka neobejdeme, jelikož nám umožňují specifikovat jednotlivé elementy při následné úpravě webové stránky v jazyce CSS. Dokument HTML je tvořen stromovou strukturou značek neboli tagů. Ty se vždy píšou do ostrých závorek a dělí se na:

- **Párové** – mají počáteční a konečný tag, mezi kterými je náležící obsah.
    - Příkladem párového tagu je například tučný text, který má počáteční tag a konečný . Text mezi těmito tagy se na stránce vypíše tučně i bez jakéhokoliv dalšího zásahu.
- **Nepárové** – stačí pouze jeden tag. Většinou se jedná o zobrazení médií.
    - Příkladem nepárového tagu je například <img>, který nám na stránce vykreslí obrázek. Aby však tento tag pracoval správně, je třeba jej obohatit o jemu náležící atributy, zejména atribut ‚src‘, který určuje cestu k požadovanému obrázku (W3C, 2014).

2.2 CSS

Kaskádové styly neboli CSS jsou doprovodným jazykem HTML, pomocí kterého řídíme vzhled stránek na koncových zařízeních. Nejedná se pouze o pozicování elementů, určování barev a animace jednotlivých elementů, ale také o tzv. Media queries, které jsou dodnes nejčastěji používaným nástrojem pro přizpůsobování webových stránek pro různá zobrazovací zařízení. V jazyce CSS převážně definujeme konkrétní elementy na stránce tím, že určujeme jejich způsob zobrazení, odsazení, rozměry a mnoho dalších vlastností, které se dále odvíjí od typu elementu (Refsnes Data, 2020).

CSS pravidla můžeme vepisovat přímo do HTML dokumentu do párového tagu <style>, nebo do elementu pomocí atributu ‚style'. Avšak mnohem přehlednější a efektivnější je zápis do externího souboru s příponou ‚.css', který poté v hlavičce HTML dokumentu zakomponujeme pomocí nepárového tagu <link> (Refsnes Data, 2020).

Základní syntaxe CSS zápisu se skládá ze selektoru a deklarací atributů s hodnotami ve složených závorkách (viz *Obr. 2*), kde je každá deklarace atributu oddělena středníkem (Refsnes Data, 2020).

```
1 .selektor {
2     atribut: hodnota;
3 }
```

Obr. 2: Definice třídy v jazyce CSS

(Zdroj: vlastní tvorba)

Pokud selektor začíná tečkou, jedná se o element s třídou selektor. Pokud by selektor začínal mřížkou (#), jedná se o element s identifikátorem selektor a v případě, že nezačíná žádným speciálním znakem, je cílen na čistý html tag (Refsnes Data, 2020).

```
1 b {
2     color: red;
3     font-size: 12px;
4 }
```

Obr. 3: Deklarování více atributů elementu v jazyce CSS

(Zdroj: Vlastní tvorba)

V případě na *Obr. 3* například říkáme HTML dokumentu, že všechny elementy <b> mají mít červenou barvu a velikost písma 12 pixelů.

2.2.1 Media Query

Na počátku přinesla media queries do tvorby webových stránek možnost specifikovat zobrazení stránek na konkrétních výstupních zařízeních dle klíčových slov pro typ médií. Mezi základní typy médií patřily například zobrazení pro televize, projektory, či handheld[8] zařízení. Nyní jsou však původní typy médií zastaralé, zejména díky rychlému rozvoji a větší různorodosti zobrazovacích zařízení. Z původních media types se zachovalo klíčové slovo ‚screen', které cílí na všechna zařízení s obrazovkou. Dále se tento typ obohacuje o tzv. media features, tedy o hlubší specifikaci koncového zobrazovacího zařízení. Pomocí media features můžeme udávat konkrétní pravidla například pro zařízení bez barevného displeje, s různými poměry rozlišení, nebo dle rozlišení v pixelech. Nejčastější metodou je stylování dle šířky prohlížeče koncového zařízení (W3C, 2017, Refsnes Data, 2020).

```
1 @media screen and (max-width: 600px) {
2     img {
3         display: none;
4     }
5 }
```

Obr. 4: Ukázka definice za pomoci media query v jazyce CSS

(Zdroj: Vlastní tvorba)

V definici v *Obr. 4* například určujeme, že na obrazovkových zařízeních, jejichž zobrazovací šířka je menší než 600 pixelů, se mají všechny obrázky (tedy elementy <img>) schovat. Pomocí parametru ‚max-width: 600px' jsme vytvořili breakpoint, který nám dává prostor pro určování pravidel, která platí právě pokud je šíře prohlížeče rovná nebo menší než 600 pixelů.

[8] Jedná se o zařízení, které se vejde do rukou

2.2.2 Použití layoutu

Jak statický, tak dynamický layout sebou nesou několik výhod a nevýhod, a proto není vždy snadné se rozhodnout pro jeden z nich. V praxi se však dle situace dá využít potenciál obou možností. Díky CSS atributu max-width můžeme udělat kompromis a využívat layout jako statický i dynamický zároveň tím, že mu nastavíme podmínky, při kterých se má zachovat jako statický a kdy jako dynamický. Pokud je šířka zobrazovacího zařízení menší, začne se layout automaticky přizpůsobovat šířce zařízení, jako by měl nastavenou šířku na 100 % (Refsnes Data, 2020).

Běžně používaná šířka hlavního layoutu je 960px. Tato šířka zůstala zvykem z dob, kdy většina návštěvníků webových stránek používala rozlišení 1024x768, a stránka tedy zahrnovala většinu obrazovky, ale po stranách byl stále prostor pro odsazení. Ačkoliv je šířka layoutu i otázkou osobních preferencí, stále se udržuje běžná šířka pro hlavní obsah někde mezi 900–1100 pixely zejména pro to, aby stránky i na velkých obrazovkách zůstaly přehledné a člověk se mohl soustředit pouze na střední část obrazovky (Knight, 2009).

2.2.3 Flexbox

Flexbox[9] je moderní technologie, která si zakládá na principu tvorby přizpůsobivého layoutu. Nabízí nám několik nových způsobů, jak na webu zarovnávat a pozicovat elementy, aniž bychom předem znali jejich velikost. Přiřazením flexboxu dáváme rodičovským[10] elementům schopnost, aby se jejich obsah uspořádal dle dostupného místa a pravidel, která můžeme dále specifikovat (Coyier, 2020).

2.2.4 CSS Grid

Na rozdíl od Flexboxu nabízí CSS Grid[11] dvourozměrný systém, který zvládá jak sloupce, tak řádky najednou. Zatímco flexbox nám bohatě vystačí pro menší využití, pomocí CSS Grid můžeme obsáhnout mnohem větší množství elementů a ovládat je najednou. Hlavní myšlenou za touto technologií je usnadnění pozicování skupin elementů a tvorby stránek založených

[9] Flexible = ohebný
[10] Jde o nadřazené elementy, také známé jako containery nebo obalovače
[11] Grid = mřížka

na mřížkových strukturách, protože ještě donedávna se pro tyto účely používaly pouze obchůzky za pomocí tabulek nebo float[12] prvků, které ne vždy odváděly dobrou práci (Coyier, 2020).

2.3 JavaScript

JavaScript (dále jen JS) je skriptovací jazyk, který můžeme stejně jako CSS implementovat do HTML dokumentů pro úpravu obsahu. Podobně jako u CSS, můžeme použít JS v dokumentu HTML přímo, a to pomocí párového tagu <script>, nebo jej zakomponovat jako externí soubor. Pomocí JS můžeme zasahovat do obsahu stránky na různých úrovních. Dle selektorů můžeme měnit jak elementy, tak jejich atributy, a tedy i jejich styly. Stylování elementů pomocí JS však není tak jednoduché jako v CSS, takže se zásahy do stylů používají většinou v případech, které nám CSS neumožňuje ovlivnit. Může se tedy například jednat o změnu jednoho elementu po kliknutí na jiný nezávislý element. Pomocí JS také můžeme ovlivňovat obsah dle šířky zobrazovacího zařízení, avšak k tomu je zapotřebí dodatečné funkce, která tuto šířku počítá a ukládá do proměnné (Refsnes Data, 2020).

Přes to všechno se však s JS ve spojitosti s responzivitou stránek můžeme setkávat velmi často. A to například při tvorbě mobilní navigace neboli tzv. ‚hamburger menu', kdy je třeba v JS nadefinovat, při jakých rozlišeních či podmínkách se má původní navigace přeměnit na mobilní verzi a umožnit správnou funkčnost mobilního menu.

2.3.1 jQuery

V návaznosti na JS je vhodné zmínit i knihovnu pro tento jazyk – jQuery. Jedná se o jednoduchou a rychlou knihovnu, která usnadňuje práci s JS. Tato jednoduchost spočívá zejména v komprimování několikařádkových JS příkazů do jednoho jQuery příkazu (Refsnes Data, 2020).

Hlavní výhodou jQuery je, že můžeme v jednom souboru prolínat jak JS, tak jQuery příkazy. Charakteristické pro jQuery je zejména selektor ‚$', který je ekvivalentem JS zápisu ‚document.querySelector' pro selekci konkrétního elementu na stránce (jQuery Foundation, 2020).

[12] Způsob řazení elementů na stránce pomocí CSS atributu ‚float'. V moderních webových technologiích se již používá velmi zřídka.

Součástí jQuery je také funkce umožňující nám sestavovat Ajax požadavky, čímž můžeme manipulovat s daty, která jsou již odeslána na serveru, a to včetně HTML struktury. V praxi můžeme vidět využití Ajaxu v prvcích stránek, ve kterých potřebujeme měnit data bez obnovování celé stránky. Pro představu si můžeme uvést například kalendář, kde díky Ajax funkci můžeme přepínat měsíce bez nutnosti obnovování celé stránky (Refsnes Data, 2020).

Knihovna jQuery volně dostupná k používání a na oficiálních stránkách můžeme nalézt i kompletní dokumentaci se všemi dostupnými funkcemi včetně příkladů použití (jQuery Foundation, 2020).

2.4 Frameworky

2.4.1 Twitter Bootstrap

Jak již název vypovídá, Twitter Boostrap je framework vytvořen vývojáři ze společnosti Twitter v druhé polovině roku 2011. V té době představil ohromnou revoluci ve formě sestaveného balíčku stylů a skriptů pro základní elementy při tvorbě webových stránek. O necelý půl rok později byl Bootstrap povýšen na verzi 2.0, která přinesla mnoho vylepšení, včetně responzivního layoutu, schopného adaptace na různá zařízení. Od té doby ušel Bootstrap pořádný kus cesty, avšak i v nejnovější verzi 4.5 využívá stále stejné principy co se týče responzivity (Cochran, 2).

Jak již bylo zmíněno, Bootstrap je volně dostupný na oficiálních stránkách, kde také můžeme najít detailní dokumentaci, příklady použití a také základní šablony. To z něj dělá vhodný nástroj i pro začátečníky v oblasti tvorby webových stránek. V dokumentaci je krok po kroku popsán postup, jak s Bootstrapem začít, včetně předpřipravených příkazů pro inkluzi všech potřebných CSS stylů i JS knihoven. Vše je doplněno o informace ohledně responzivity a uživatel je také seznámen se všemi součástmi a jejich funkcemi (Twitter, 2020).

Koncept Bootstrapu si zakládá na mřížkovém systému, který spočívá v rozdělení webových stránek do 12 pomyslných sloupců, které jsou součástí tzv. containeru obsahu stránky. Tyto containery jsou základem, bez kterého systém sloupců nezprovozníme. Na výběr máme hned z šesti možností, které se odvíjí od maximální velikosti, kterou potřebujeme. Žádný z nich však nepřesahuje šířku 1140 pixelů a liší se zejména v tom, od jaké velikosti zařízení se přizpůsobuje prohlížeči. Jinými slovy určujeme, za jakých podmínek se tento container chová adaptivně a kdy

responzivně, přičemž nejjednodušší je základní container označený třídou ‚container', který se chová adaptivně dle různých breakpointů rozdělených na extra malá, malá, střední, velká a extra velká zařízení. Při zařazování obsahu do containeru přiřazujeme jednotlivým elementům třídu ‚col' (zkráceně pro column = sloupec) společně s počtem sloupců, které má tento element zabírat. V praxi to tedy vypadá tak, že třísloupcový element zabírá třetinu containeru, zatímco šestisloupcový zabírá polovinu atp. (Twitter, 2020).

Jelikož je Bootstrap charakterizován jako mobile first framework, nemělo by nás překvapovat, že základní kostra media query v CSS je seřazena vzestupně podle velikosti. To však nevylučuje možnost přeskupení a následného vytváření pravidel sestupně.

Součástí Bootstrapu jsou také JS pluginy[13] jQuery a Popper.js. Obě tyto knihovny mají za úkol usnadňovat nám nejen tvorbu responzivních stránek, ale také implementaci nejčastěji používaných elementů, jako jsou modální okna pro upozornění, nebo karusel (slider). Veškeré závislosti obsažených funkcí na odpovídajících knihovnách jsou samozřejmě dostupné k zobrazení na oficiálních stránkách dokumentace Bootstrapu (Twitter, 2020).

2.4.2 ZURB Foundation

Podobně jako Twitter Bootstrap, Foundation je jedním z předních frameworků pro tvorbu responzivních stránek na současném trhu, zakládajících si na systému mřížky. Stejně tak nabízí možnost tvorby webu přístupem mobile first či desktop first. Oproti Bootstrapu se však Foundation jeví jako o něco složitější a komplexnější, což není na škodu v případě, že již máme nějaký přehled o tvorbě responzivních webových stránek a chceme si vytvořit stránky šité na míru. Na druhou stranu nabízí Foundation mimo online dokumentace i video tutoriály nebo placené kurzy, vhodné i pro začátečníky. Mimo předem sestavených šablon nabízí Foundation i jednotlivé bloky elementů, pomocí kterých si můžeme stránku posléze poskládat (ZURB, 2020; Genge, 2018).

Princip responzivity za pomocí frameworku Foundation je také založen na principu mřížky. Avšak v posledních letech se stále vyvíjí a od verze 6.4 (nejnovější verzí ke dni psaní práce je 6.6) představil tzv. XY Grid, který svým způsobem nahradil původní Float Grid a Flex Grid. Avšak

[13] Plugin = softwarový doplňek.

co se týče praktického využití, podobnost s Bootstrapovou mřížkou a principem rozdělování elementů do sloupců je téměř identická (ZURB, 2020).

2.4.3 AMP

AMP je moderní framework, který přináší do HTML novou unikátní syntaxi a jeho hlavním záměrem je vytváření uživatelsky přívětivých stránek, které se načítají téměř instantně. Rychlost je hlavním cílem AMP a dosahuje jí několika způsoby. Základním principem AMP je současné načítání všech komponentů webu a omezování JS třetích stran, které často zdržují proces načítání. Také dovoluje pouze jedny kaskádové styly, které jsou omezeny velikostí 50 kilobajtů. Hlavní dominantou AMP je načítání stránky dopředu (kešování) na odděleném serveru. Na tomto serveru probíhají výpočty, které následně optimalizují zobrazení stránky pro cílové zařízení. Tato optimalizace je zejména cílena na minifikaci[14] kódu a nahrazování médií jinými, méně náročnými formáty (OpenJS Foundation, 2020).

Nevýhodou však může být například omezená možnost stylování a úpravy stránek na míru kvůli dříve zmíněnému limitu velikosti CSS a specifické syntaxi jazyka AMP HTML. Několik zdrojů také vidí potencionální hrozbu v uskupování velkého množství stránek pod jednou společností, což by mohlo vést ke snadnému zneužití některých stránek například ve formě snadné distribuce falešných informací, nebo zranitelnosti vůči útokům. Tato varianta však není zcela proveditelná, protože pokud by se všechny stránky kešovaly, velikost internetu by se zdvojnásobila (Gilbertson, 2017; Sheffield, 2017).

2.5 Vývojová prostředí

V souvislosti se zmíněnými moderními technologiemi a tvorbou layoutu v této kapitole je velmi důležitá volba vhodného vývojového prostředí.

Vývojové prostředí (dále IDE), slouží jako základní nástroj pro každého programátora. Ačkoliv je možné naprogramovat webové stránky v poznámkovém bloku nebo jiném textovém editoru, je mnohokrát lepší si zvolit nějaké IDE, které nám práci usnadňuje hned několika

[14] Minifikace = minimalizace kódu; jde o proces odmazávání přebytečných mezer a odsazení v kódu tak, aby zůstal stále funkční, avšak zabíral co nejméně místa v úložišti.

funkcemi. To nám totiž dovoluje mnohem více než pouze psát a upravovat kód (Codecademy, 2020).

Jednou ze základních funkcí je například zvýrazňování syntaxe. Tato funkce barevně rozděluje různé elementy zvoleného jazyka, čímž činí kód mnohem přehlednější (viz *Obr. 5* a *Obr. 6*). Špatným vybarvením můžeme také například zjistit, že jsme použili špatný syntaktický zápis, nebo máme někde překlep (Microsoft, 2020).

```
1  <!-- Ukázka kódu bez zvýrazňování syntaxe -->
2
3  <html>
4      <div class="test">
5          <p>lorem ipsum dolor sit amet</p>
6      </div>
7  </html>
8
```

Obr. 5: Kód bez zvýraznění HTML syntaxe

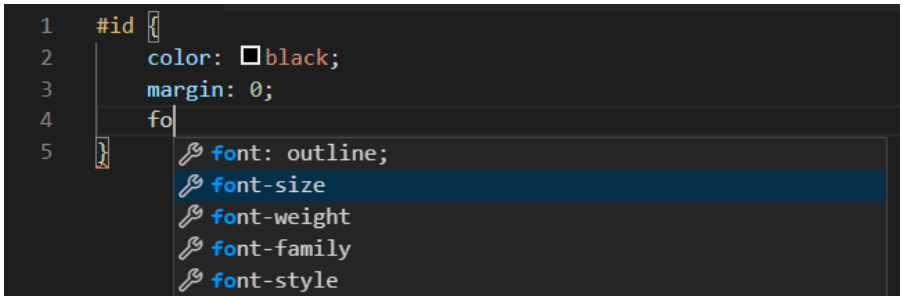
(Zdroj: vlastní tvorba)

```
1  <!-- Ukázka kódu se zvýrazňováním syntaxe -->
2
3  <html>
4      <div class="test">
5          <p>lorem ipsum dolor sit amet</p>
6      </div>
7  </html>
8
```

Obr. 6: Kód se zvýrazněnou HTML syntaxí

(Zdroj: Vlastní tvorba)

Další funkcí, kterou je vhodno zmínit, je funkce našeptávání a automatického dokončování. V tomto případě nám IDE během psaní nabízí možnosti, které v daném jazyce můžeme použít. Tato funkce není vhodná jen na rychlé dokončování syntaxe, ale také nám pomáhá zjistit správnou formu zápisu a zda v této chvíli opravdu můžeme příkaz použít, jak je vidno v *Obr. 7*. V dnešní době se tyto funkce také uskupují a využívají pod názvem IntelliSense pro daný jazyk (Microsoft, 2020).



Obr. 7: Funkce automatického doplňování a našeptavač v IDE Visual Studio Code

(Zdroj: Vlastní tvorba)

Pro účely této práce se nebudeme příliš zabývat dalšími funkcemi, které může IDE nabízet, jelikož jich je nepřeberné množství a při tvorbě jednodušších webových stránek je všechny nevyužijeme. Každý si však pro určitý programovací jazyk a použití najde své výhody a IDE, které mu nejvíce vyhovuje.

Mezi nejpoužívanější IDE patří například NetBeans, Eclipse nebo Visual Studio Code (Carbonnelle, 2020).

2.6 Modulové a grafické technologie pro tvorbu webdesignu

Často bývá zvykem, že se před programováním stránek vytvoří nejprve grafický návrh, který se dle požadavků a připomínek mění do své finální podoby.

Stále známějším se však stává software, ve kterém si můžeme sami vizuálně poskládat stránku z jednotlivých modulů, vybrat si šablonu a v některých případech ji dokonce můžeme rovnou přes tento software umístit na internet. Toto řešení se sebou však nese několik výhod a nevýhod. Tyto tzv. website buildery můžeme nalézt většinou jako online nástroje, ať už zdarma s omezeními, nebo placené. Mezi tyto website buildery patří například Wix.com [15] nebo Webnode[16]. Všechny tyto služby argumentují zejména tím, že při tvorbě stránky pomocí jejich služby nepotřebujete umět žádný programovací jazyk ani se orientovat v profesionálních grafických programech, což samozřejmě může být lákadlem pro menší a začínající podnikatele či bloggery (Escalona, 2017).

[15] Dostupné z: www.wix.com
[16] Dostupné z: www.webnode.com

Na druhou stranu se však stále více vyplatí tvorba stránek šitých na míru, na což website buildery nemají dostatečně mnoho funkcí a možností úprav. Seriózní klienti chtějí mít své stránky originální a s šablonami, které používají tisíce dalších webů, se jednoduše nespokojí. V této chvíli přichází na scénu poctivější tvorba webdesignu. Při tvorbě webového designu totiž nejde pouze o naskládání obsahu tak, jak se nám zlíbí, ale je třeba brát v potaz i marketingové aspekty (Escalona, 2017).

Designeři často využívají i principy psychologie a marketingu, aby ušili obsah a vzhled webu na míru odpovídající jejímu záměru a cílové skupině. Při tvorbě vzhledu se často používají programy jako Sketch, který slouží pro blokový návrh i s možnými interakcemi, nebo Adobe Photoshop, ve kterém se vytváří konečný grafický podklad pro kodéra. Ze výstupních souborů těchto programů navíc může kodér vyčíst i konkrétní CSS vlastnosti, aby mohl snadněji vytvořit věrnou reprezentaci návrhu (Sergey L., 2018).

Nyní, když jsme si představili nabídku využitelných moderních technologií v oblasti tvorby webových stránek a jejich praktické využití, se dále přesuneme k problematice přístupů k tvorbě webdesignu.

3 Přístupy k tvorbě webdesignu

V této kapitole si rozebereme možné přístupy, kterými se můžeme při tvorbě webových stránek řídit. Jedná se zejména o filozofické směry, na které později aplikujeme praktická využití výše zmíněných moderních technologií.

3.1 Desktop first

Desktop first je svým způsobem klasický přístup, který známe a dodnes se používá. Celý spočívá v tom, že webovou stránku nejprve navrhujeme pro desktopová zařízení s velkými rozlišeními, a mobilní verzi dále vytváříme „osekáváním“ desktopové verze. Tato metoda však může být v dnešní době ošemetná, protože při tvorbě webu pro velké rozlišení je snadné se nechat unést efekty a velkými dávkami informací, zatímco při tvorbě mobilní verze musíme zvážit, které aspekty zanechat, které schovat a jak stránky uzpůsobit, aby mobilního návštěvníka zaujaly (Kaempf, 2018).

Zde se také dostáváme k problému rozdílu adaptivity a responzivity. Informační zdroje uvádí, že tento rozdíl je zejména ve filozofii a detailech. V praxi však jeden druhý nevylučuje. Responzivita webu zaručuje, že šířka webové stránky se za všech okolností přizpůsobuje prohlížeči. Na rozdíl od adaptivity však neřeší obsahovou stránku webu. Zde tedy přichází na řadu adaptivita, která má za úkol zejména přizpůsobování stránek na konkrétní zařízení za pomocí dříve zmíněných breakpointů, kdy nejen přizpůsobuje stránku prohlížeči, ale také se zabývá úpravou struktury obsahu stránky včetně typografických pravidel. V praxi se však setkáváme s kombinací obou stylů. Zatímco adaptivita nám poskytuje uživatelsky přívětivou prezentaci stránek na různých zařízeních, responzivita zaručuje (v případě správného provedení) správné zobrazování napříč různými rozlišeními, a potencionálně může stránkám zaručit bezproblémový chod i na nových obrazovkových zařízeních, která přijdou v blízké budoucnosti (Graham, 2015).

3.2 Mobile first

Jelikož mobilní zařízení nyní převládají statistiky návštěvností webu (viz *Obr. 1*), bylo třeba uzpůsobit i přístupy tvorby stránek tak, aby tato cílová skupina byla na primárním místě.

S tímto záměrem vznikl i termín mobile first, který přivedl na svět v roce 2009 Luke Wroblewski v jeho stejnojmenné knize. Princip mobile first spočívá v tom, že nejprve navrhneme vzhled a rozložení stránky pro mobilní zařízení, a místo vytváření media query breakpointů na postupně nižší rozlišení naopak postupujeme vzestupně. Díky tomuto přístupu již od počátku rozhodujeme, které elementy stránky jsou důležité a zajímavé pro návštěvníky (Wroblewski, 2009; Kaempf, 2018).

Nejde však jen o vzhled, musíme také brát v zřetel vytížení stránek, jelikož uživatelé počítají s tím, že na mobilních zařízeních se stránka načte rychle a nebude spotřebovávat mnoho dat. Již ze statistik v roce 2016 jsme se dozvěděli, že více než polovina návštěvníků opouští stránky již po třech sekundách načítání. Kamenem úrazu v případě spotřeby dat bývají zejména videa a obrázky na stránkách. Zde je například potřeba dávat pozor na schovávání elementů pomocí CSS příkazu ‚display: none'. Tento příkaz totiž sice schová daný element, avšak ten se v některých prohlížečích stejně načte a spotřebuje data zbytečně (Everts, 2016; Wroblewski, 2014).

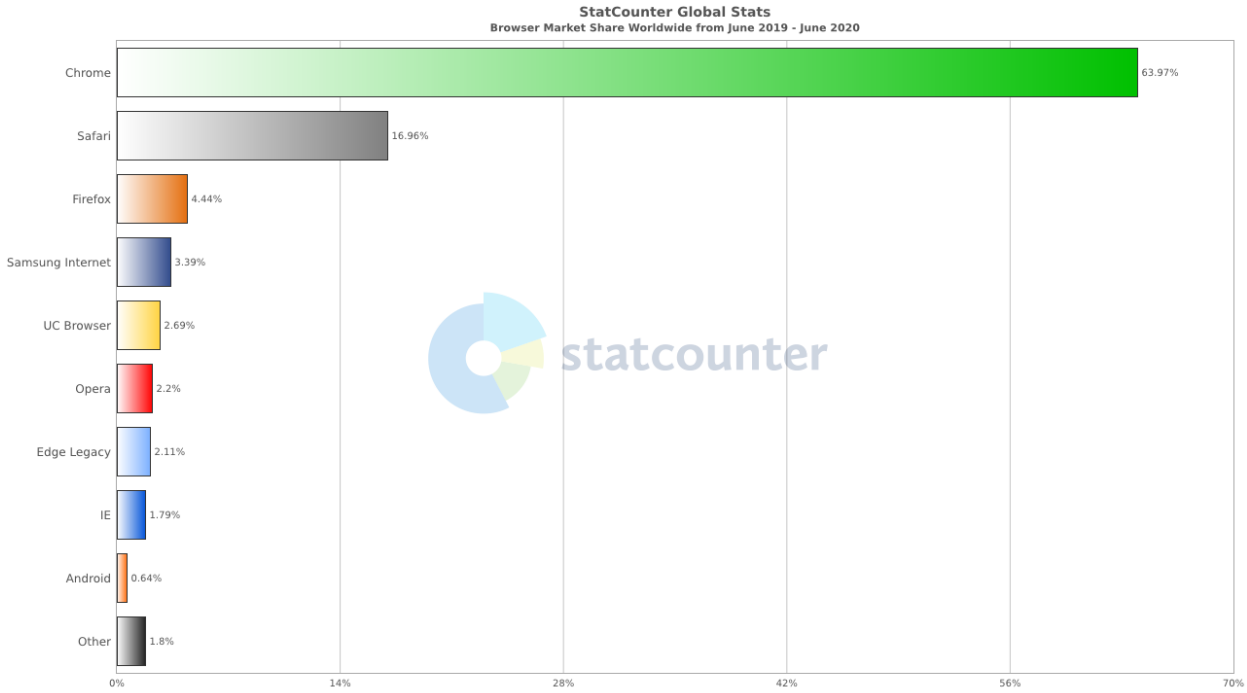
Dle dostupných zdrojů zabývajících se přístupy k tvorbě stránek není jednoduché rozhodnout, zda je lepší jeden nebo druhý způsob. Jakmile začneme mobilním designem, musíme počítat s tím, že desktopová verze nebude perfektní a naopak. Různé zdroje se také rozchází v terminologii a vylučují kombinaci responzivního a mobile first designu. Nutno však podotknout, že vhodný výběr přístupu ke tvorbě webu záleží na obsahu. Pokud chceme stránky s obsáhlými texty a poutavým vizuálem, mobile first není vhodnou volbou, jelikož využívá zejména jednoduchosti a stručnosti. Příliš mnoho informací a přemrštěný vzhled může mobilního návštěvníka jednoduše odradit (Kaempf, 2018).

V mnoha případech jde při volbě přístupu tvorby webu o bezpočet faktorů. Žádný z těchto směrů však není charakterizován jako nutnost a konečný přístup může nakonec být otázkou pouhé osobní preference.

4 Klientská strana

Hlavní cíl bakalářské práce je zaměřen zejména problematiku responzivního webdesignu. Při tvorbě responzivních webových stránek není důležité jen to, co je na serveru. Velmi důležité je také dbát na to, jak ke stránkám přistupují uživatelé (klienti). Z této strany musíme brát ohled nejen na zařízení a rozlišení, ale také na webové prohlížeče. Jelikož tvoříme stránky zejména pro to, aby je lidé navštěvovali, musíme se koncovým uživatelům přizpůsobovat a stavět stránky tak, aby vyhovovaly co největšímu množství zařízení. Dále se budeme tedy zabývat responzivním webdesignem z pohledu uživatele (klienta).

4.1 Webové prohlížeče



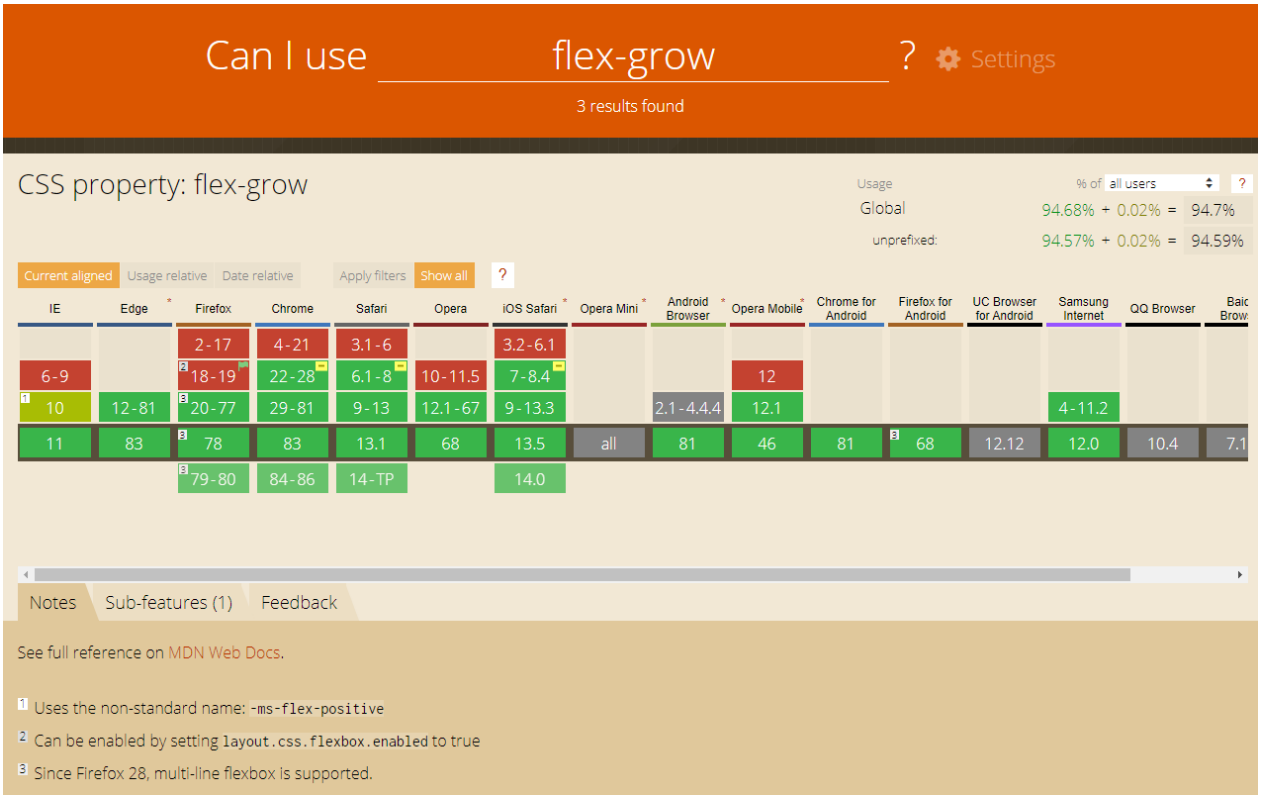
Obr. 8: Oblíbenost prohlížečů za poslední rok

(Zdroj: Statcounter, 2020)

Další věcí, na kterou je třeba si dávat pozor při tvorbě stránek je kompatibilita s prohlížeči. Na *Obr. 8* můžeme vidět popularitu různých prohlížečů za poslední rok. Často se setkáme se situacemi, kdy nemusíme koncové prohlížeče vůbec řešit. Dříve jsme se setkávali s problémy způsobenými tím, že každý prohlížeč má pro jednotlivé HTML elementy nastaveny původní CSS

hodnoty jinak. Jedná se většinou o různé výšky řádku textu, velikosti nadpisů, nebo odsazení. Pro tyto potřeby však vznikly knihovny jako například Reset.css a Normalize.css, které přepisují a sjednocují základní styly všech prohlížečů. Dnes bývají tyto knihovny již zakomponovány ve frameworcích a větších knihovnách – například Reboot, založený na Normalize.css, je již základní součástí dříve zmiňovaného Twitter Bootstrap frameworku (Meyer, Meyer, 2020; Coyier, 2017; Gallagher, 2012).

Problém, se kterým se však setkáváme dnes a denně, je kompatibilita prohlížečů s novými, a ještě nezaběhlými technologiemi. Běžné statistiky oblíbenosti prohlížečů nám bohužel nesdělí, jaké konkrétní verze uživatelé používají, a musíme tedy brát ohled i na zařízení, které využívají zastaralé verze prohlížečů a buďto některé funkce vynechat, nebo najít způsob, jak je co nejvěrněji interpretovat pro zastaralé prohlížeče. Pokud se tedy chystáme použít novější a netradiční postupy, přijdou nám vhod online stránky, které nám sdělí, na jakých verzích a s jakými obtížemi můžeme námi zvolenou technologii použít (Refsnes Data, 2020; Coyier, 2017).



Obr. 9: Ověření kompatibility CSS atributu flex-grow s prohlížeči

(Zdroj: Deveria, 2020)

Na *Obr. 9* můžeme vidět, jakým stylem si můžeme ověřit kompatibilitu pomocí stránky *Can I use... Support tables for HTML5, CSS3, etc*[17]. Tato stránka nám vypíše detailní souhrn všech podporovaných (zelených) i nepodporovaných (červených) verzí prohlížečů, společně s procentuálním podílem jejich využití na trhu, abychom si mohli odvodit, zda se jedná o zanedbatelné množství návštěvníků či nikoliv. Společně s těmito informacemi dostaneme i rady, jak je možné problémy s kompatibilitou v některých případech obejít, i s odkazy na dokumentaci vztahující se k hledanému výrazu (Deveria, 2020).

Při problémech týkajících se kompatibility s CSS atributy, můžeme stále narazit na pomocné předpony neboli prefixy, které pomáhají ojedinělé případy vyřešit. Tyto prefixy pomáhají zprovoznit vybrané funkce i na zastaralých prohlížečích. Jedná se zejména o následující:

- ‚-ms-, pro podporu na prohlížeči Internet Explorer/Edge,
- ‚-moz-, pro podporu na prohlížeči Mozilla Firefox,
- ‚-webkit-, pro podporu na prohlížečích Google Chrome, Safari a Opera (Refsnes Data, 2020).

4.2 Přístupnost

Od roku 2019 je díky Zákonu o přístupnosti webových stránek a mobilních aplikací č. 99/2019 Sb. přístupnost webových stránek brána mnohem vážněji. Tento zákon stanoví, že stránky a aplikace, patřící pod povinný subjekt, musí být kompletně přístupné pro osoby se zdravotním postižením.

Jako příklad si můžeme uvést přístup stránek pro návštěvníky využívající odečítače obrazovky, které interpretují webové stránky ve formě předčítání nebo převodu na Braillovo písmo. Na špatně přístupných stránkách se lidé, kteří používají odečítače obrazovky, mohou potýkat s následujícími problémy:

- Vizuální elementy bez popisu nebo alternativního textu
- Funkční prvky bez možnosti ovládání klávesnicí
- Špatně strukturovaný obsah a navigace

[17] Dostupné z: www.caniuse.com

- Neočekávané a časované akce – např. přesměrování na jinou stránku po skončení videa (Digital Education Strategies, The Chang School, 2019).

Nejde však pouze o speciální zařízení. Návštěvníci s poruchami zraku mohou také využívat programy pro zvětšení textu nebo změnu barev. V těchto případech také čelíme mnoha potencionálním problémům, na které je třeba dbát již při kódování stránek. Mezi běžné problémy a jejich řešení patří:

- Nevhodné použití velikostí – zásadním problémem je nastavování pevných velikostí. Je proto vhodné uvádět každý text a odsazení v jednotkách em, rem, nebo %.
- Špatný kontrast barev – pro osoby se slabším zrakem nebo barvoslepostí mohou některé barevné kombinace být nečitelné, je tedy dobré zvolit dostatečně kontrastní barvy jak při výběru textu, tak pozadí. Problémy mohou nastat i pokud použijeme efekt změny barvy textu jako funkční prvek.
- Zvolení stylu písma – některé styly písma se sebou nesou v případě přístupnosti řadu problémů. Při výběru písma bychom měli dbát zejména na čitelnost. Vhodné tedy je vynechat přehnaně dekorativní nápisy nebo příliš tenká písma, která i po zvětšení nemusí být dobře čitelná (Digital Education Strategies, The Chang School, 2019).

Naštěstí máme při tvorbě webových stránek k dispozici mnoho evaluačních nástrojů, které na cílové stránce bezplatně vyhodnotí míru přístupnosti stránek a okamžitě informují, které věci je třeba napravit či vylepšit, jako například *WAVE Web Accessibility Evaluation Tool*[18].

4.3 Zobrazovací zařízení

Kvůli technologickému rozvoji a vzniku nových zařízení, které slouží také pro zobrazování webových stránek, bylo třeba vytvořit již dříve zmíněné Media query. Tato funkce spatřila světlo světa již v CSS2 ve spojitosti s HTML4, které přineslo možnost cílení kaskádových stylů stránek na konkrétní zařízení. Atribut ‚media' v HTML4 nám dodal možnost zaměřit se na širokou škálu dostupných zařízení:

[18] Dostupné z: www.wave.webaim.org

- braille – cíleno na zařízení interpretující stránky v Braillově písmě;
- embossed – pro tiskárny umožňující tisk v Braillově písmě;
- handheld – určeno pro „ruční“ zařízení s malými obrazovkami a omezenou šířkou pásma;
- print – cíleno na stránkový tisk materiálů a dokumentů a jejich správné zobrazení v náhledovém režimu pro tisk;
- projection – prezentace stránek, zejména určeno pro projektory;
- screen – primárně zaměřeno na zařízení s barevnými obrazovkami;
- speech – cíleno na zařízení předčítače stránek. V CSS2 byla uvedena obdoba tohoto typu – aural;
- tty – určeno pro zařízení s pevně danou mřížkou pro znaky. Jednalo se zejména o terminály nebo přenosná zařízení s velice omezeným displejem. Při stylování tohoto typu nebylo vhodné využívat jednotky pixelů;
- tv – pro televizní zařízení, která nabízela zvuk a omezené možnosti posouvání stránek;
- all – používáno pro cílení všech zařízení (W3C, 2017).

Mnoho z těchto typů však ve výsledku u koncových zařízení nenašlo podporu a zachovaly se pouze některé z nich – all, print, screen a speech. Místo zaměření se na konkrétní zařízení se tedy začala Media query ve verzi CSS3 obohacovat o obecnější možnosti zaměřování se na koncová zařízení. Místo toho, abychom určili konkrétní koncové zařízení, zabýváme se spíš jeho schopnostmi a vlastnostmi. Hlavními hodnotami, na které se v případě media query zaměřujeme, jsou například rozlišení a orientace zařízení. Ve specifikaci media query však můžeme najít i velmi specifická pravidla, která lze v ojedinělých případech použít. Mezi ty patří například barevná hloubka, poměr stran, úroveň osvětlení, nebo dokonce preference uživatele ohledně kontrastu a barevného schématu (Refsnes Data, 2020; W3C, 2017).

5 Praktická část

Dílčím cílem bakalářské práce je představit moderní webové technologie a přístupy při tvorbě responzivního webdesignu. V praktické části této práce se zaměříme na hlavní cíl, a to aplikaci moderních webových technologií zmíněných v teoretické části (viz kapitola 2) na neresponzivní webové stránky za účelem zlepšení jejich responzivity.

Pro potřeby této práce jsme se rozhodli přepracovat stránky firmy pro prodej palivového dřeva v Horním Benešově, které jsou dostupné na adrese www.drevokuchar.cz. Pro možnost porovnání jsme na subdoméně www.old.drevokuchar.cz nechali starou verzi, kterou tvořil autor práce před několika lety a nyní je již zastaralá a nevyhovuje moderním standardům. Jedná se o tzv. one-page, tedy jedinou stránku sloužící pouze pro prezentační a informační účely. Rozhodli jsme se při tvorbě používat IDE Microsoft Visual Studio Code. Při výběru IDE rozhodovalo více faktorů. Zejména se jednalo o jeho dostupnost ve verzi freeware, tj. volně stažitelný a také jeho nabídku nepřeberného množství funkcí a možností přizpůsobení. I když tyto stránky nepatří pod veřejný subjekt, budeme se v mnoha ohledech snažit dodržovat i pravidla přístupnosti webových stránek kvůli lepší reprezentaci a cílení na širší skupiny potencionálních návštěvníků. Kvůli jednoduchosti stránek jsme se také rozhodli nepoužívat žádný framework, abychom udržovali kód jednoduchý, čitelný a neměli problémy s přizpůsobením jednotlivých elementů.

```html
<!DOCTYPE html>
<html lang="cs">

<head>
    <title>Prodej palivového dřeva v Horním Benešově</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8">
    <meta name="keywords" content="palivové dřevo, prodej dřeva">
    <meta name="description" content="Prodej palivového dřeva v okrese Bruntál a okolí. ">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="robots" content="index,follow">
    <meta content="IE=edge" http-equiv="X-UA-Compatible">

    <link rel="icon" type="image/png" href="img/favicon.png" />
    <link href="css/styles.css" rel="stylesheet">
</head>

<body>

    Zde bude obsah stránky

    <script src="js/lightbox-plus-jquery.min.js"></script>
    <script src="js/script.js"></script>
</body>

</html>
```

Obr. 10: Počáteční nastavení HTML kódu

(Zdroj: Vlastní tvorba)

Na *Obr. 10* můžeme vidět základní nastavení webové stránky před kódováním jejího obsahu. V hlavičce stránky definujeme většinu informací, které nejsou na stránce nijak očividné, ale poskytují informace pro zařízení a online vyhledávače. Tyto informace se ukládají do tzv. meta tagů a mezi základní patří tyto:

- **Content-type** – druh obsahu, zde určujeme, že jde o html soubor a také definujeme znakovou sadu. Pro češtinu používáme znakovou sadu ‚utf-8‘
- **Keywords** – klíčová slova webu, slouží zejména pro vyhledávače. Zde zadáváme zkrácené výrazy a fráze spojené s obsahem stránky
- **Description** – popis stránky, tento popis se zobrazuje ve výsledcích vyhledávání pod názvem stránky
- **Viewport** – v současné době jde o velmi důležitý údaj, který nám umožňuje tvorbu responzivního webu. Pomocí tohoto tagu diktujeme, aby se šířka stránky přizpůsobila šířce zařízení, které si web zrovna prohlíží. Také jím nastavujeme počáteční škálování na 100%

- **Robots** – instrukce pro internetové vyhledávače. Původní hodnota je ‚index', což umožňuje zahrnování našeho webu do online vyhledávačů. Tuto možnost můžeme i vypnout zadáním hodnoty ‚noindex'
- Poslední meta tag s obsahem ‚IE=edge' nám dovoluje podporu webu na starších verzích prohlížeče Internet Explorer.

Zbytek základního nastavení (kromě tagu <title>, který určuje titulek naší stránky) se zabývá hlavně propojením souboru index.html s dalšími pomocnými soubory, které budeme používat. Jedná se hlavně o CSS a JS soubory. Zatímco kaskádové styly definujeme v hlavičce, naopak JS soubory je vhodné připojit až na konci souboru. Tím zabráníme problémům se zobrazováním stránek v případě chybných skriptů, jelikož hierarchická posloupnost nám v případě chyby dovolí načíst základní stránku a kaskádové styly i když na konci zjistí, že JS soubory načíst nelze – v tomto případě tedy můžeme přijít o některé funkcionality jako například zobrazení galerií nebo některé vizuální efekty, avšak stránky budou stále funkční a veškeré informace dosažitelné.

V případě této stránky jsme se rozhodli zahrnout mimo vlastního JS také knihovnu *Lightbox*[19], která nám usnadní práci s galerií a zobrazováním fotek.

Společně s počátečním nastavením HTML struktury jsme zakomponovali i základní CSS styly. Jednalo se zejména o připojení jiných stylů, které využijeme pro dříve zmíněnou knihovnu Lightbox.js, implementace fontu z *Google Fonts*[20] a vymazání základních stylů, které si určují prohlížeče, abychom mohli veškerá odsazení a velikosti stylovat od nuly, v libovolné míře a pro všechny prohlížeče stejně. Dle pravidel přístupnosti webu jsme také definovali základní velikost písma na 100 %, čímž se přepočítává základní písmo dle velikosti, kterou má uživatel nastavenou v prohlížeči (viz *Obr. 11*).

[19] Dostupné z: www.lokeshdhakar.com/projects/lightbox2
[20] Dostupné z: www.fonts.google.com

```css
/* Implementace */
@import url('https://fonts.googleapis.com/css2?family=Raleway:wght@400;500;600;700&display=swap');
@import 'lightbox.min.css';

/*
*   Globální nastavení
*/

* {
    font-family: 'Raleway', sans-serif;
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    scroll-behavior: smooth;
}

body {
    font-size: 100%;
    padding-left: 15%;
}

a {
    text-decoration: none;
    color: #E4CDA3;
}
```

Obr. 11: Počáteční nastavení kaskádových stylů

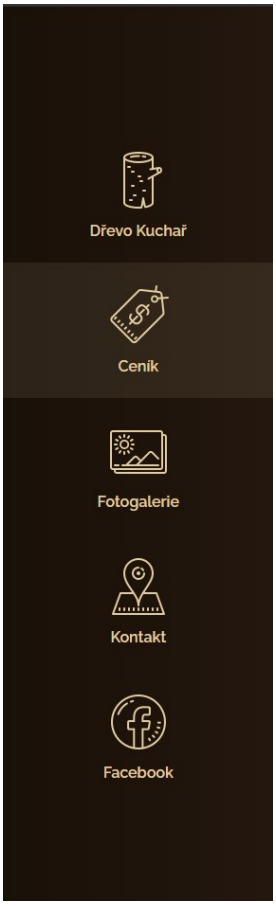
(Zdroj: Vlastní tvorba)

Hlavní navigace

Po počátečním nastavení webu jsme začali přípravou hlavní navigace. Rozhodli jsme se použít fixní navigaci po levé straně nejen z důvodu originality, ale také nám pro tento druh stránky přišla vhodnější. Z počátku jsme chtěli navigaci udělat úplně jednoduchou – pouze jako seznam textových odkazů, ale v tomto případě bylo po horní a spodní straně menu příliš mnoho volného místa a menu nepůsobilo tak výrazně. Rozhodli jsme se tedy tyto texty obohatit ikonkami, které daly navigaci trochu jiný nádech a vyplnily prázdná místa. Ikonky jsme stáhli zdarma z portálu *Icons 8 - Download free icons, music, stock photos, vectors*[21], avšak pod podmínkou, že na stránkách v patičce zanecháme odkaz na jejich portál, aby nedošlo k porušení autorských práv. Při těchto úpravách jsme uvítali absenci menu z některých frameworků, protože například Bootstrapové menu obsahuje velké množství tříd a původních definic a změna vzhledu takovéhoto menu by zabrala mnohem více času. Jako pozadí jsme zprvu zvolili čistou jednotnou barvu, ale aby působila trochu více zajímavěji, využili jsem online nástroj *Ultimate CSS Gradient*

[21] Dostupné z: www.icons8.com

Generator - ColorZilla.com[22], pomocí kterého jsme vytvořili lehký přechod, který je trochu znatelný, ale zároveň nepůsobí přehnaně. Tento proces jsme poté použili znovu u bloků s ceníkem a kontaktem, které původně měly mít jednobarevné pozadí.



Obr. 12: Hlavní navigace
(Zdroj: Vlastní tvorba)

Jednotlivá tlačítka nejsou zpracována klasicky jako seznam, ale kvůli lepší funkcionalitě jde o odkazy, které jsme v CSS pomocí flexboxu seřadili do sloupce a zarovnali na střed jak horizontálně, tak vertikálně. Na konec jsme přiřadili každému odkazu efekt zesvětlení za pomocí pseudo-elementu ‚after', který se při najetí na odkaz objeví (viz *Obr. 12*) Šířku menu jsme ze základu nastavili na 15 %, stejně tak jako odsazení těla stránky z leva, aby nedocházelo

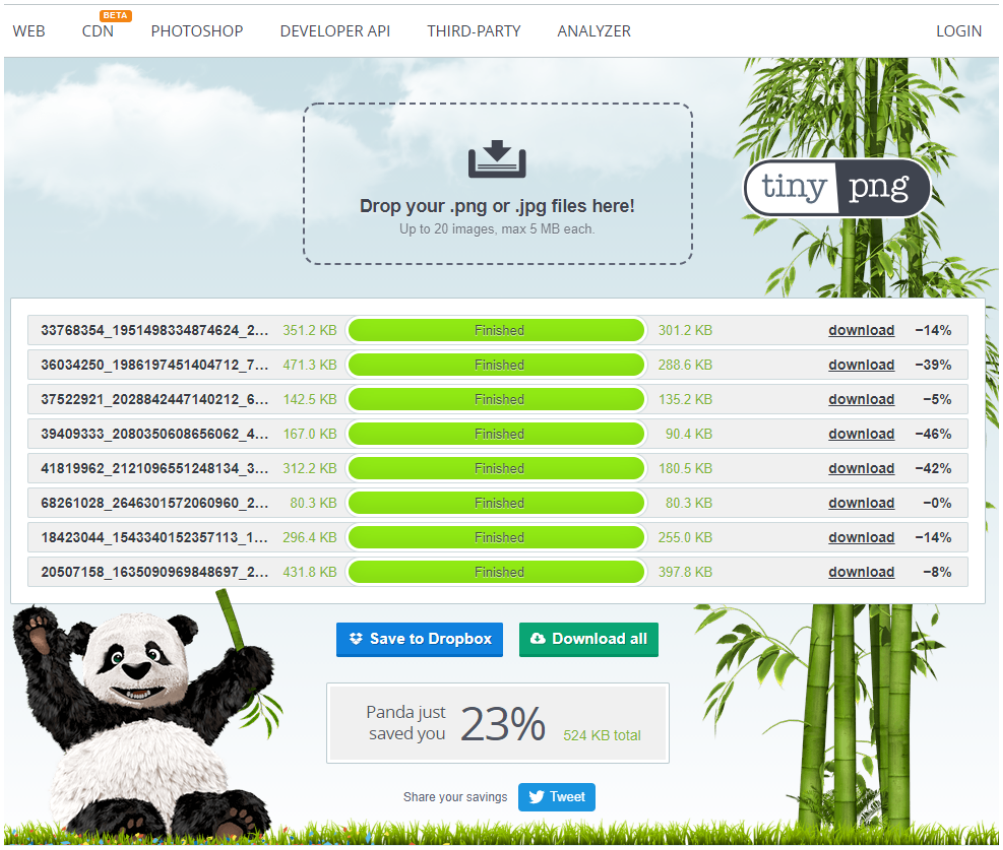
[22] Dostupné z: www.colorzilla.com/gradient-editor

k překrývání s obsahem. Responzivitou menu, stejně jako ostatních elementů, se budeme zabývat až na konci, abychom mohli testovat a ladit vše najednou.

Obsahové bloky

Při tvorbě obsahu jsme chtěli zachovat původní strukturu, tedy úvodní blok, ceník s informacemi, galerii a kontakt s mapou. Většinu věcí jsme si tedy byli schopní nachystat dopředu s využitím vědomostí autora této práce, nabytých za poslední roky. Jelikož byla naším hlavním cílem lepší responzivita webu, předem jsme pomocí online nástroje *TinyPNG – Compress PNG images while preserving transparency* [23] optimalizovali všechny obrázky použité na webu, aby návštěvník nepřicházel o drahocenná data navíc. Jedná se o jednoduchý nástroj, který komprimuje obrázky, aby zabíraly nejméně místa, zatímco si zachovaly kvalitu. V našem případě jsme tímto ušetřili zhruba 23 % místa jen na galerii (viz *Obr. 12*).

[23] Dostupné z: www.tinypng.com



Obr. 13: Online nástroj pro optimalizaci obrázků

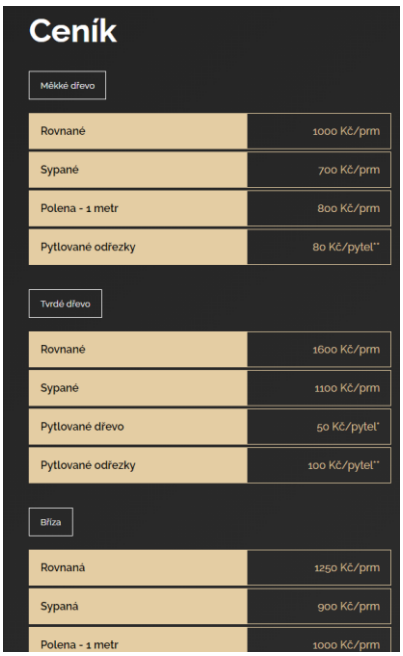
(Zdroj: Voormedia, 2020)

Poté jsme začali úvodním blokem, který jsme se rozhodli protáhnout nejen na plnou šířku, ale také plnou výšku zařízení, abychom docílil efektu prezentačního webu a zároveň si zajistili dostatek místa pro úvodní informace v případě, že by se v budoucnu ještě doplnily nebo změnily. Zde byl pouze problém s výpisem textu na obrázku. V těchto případech nastávají problémy často kvůli zachování dobré čitelnosti textu. Rozhodli jsem se tedy zastřít obrázek v pozadí tzv. divem[24], který zaplňuje celou plochu bloku, má barevné pozadí a sníženou průhlednost. Pro případ, že by někteří uživatelé používali prohlížeč, který nepodporuje průhlednost, jsme ještě dodatečně aplikovali menší stín na text, aby byla zachována alespoň nějaká čitelnost.

Dalším na řadě byl blok s ceníkem a informacemi. Zde jsme se potýkali s mnoha problémy a složitým plánováním. Chtěli jsme udělat ceník interaktivnější a přehlednější, aby si uživatel mohl přepínat mezi možnostmi a nemusel se orientovat v rozsáhlé tabulce, jako tomu bylo doposud.

[24] Univerzální párový tag .

Věděli jsme však, že k tomu potřebujeme použít JS, ale zároveň jsme chtěli ceník zachovat přístupný pro zařízení, které JS nepodporují. Mohli jsme celý blok zkopírovat a nastylovat jej zvlášť, ale chtěli jsme se vyhnout duplicitnímu kódu na stránce, a proto jsme se pokusili nechat jen jeden blok a rozdělovat jej stylováním. Toho jsme docílili tím, že jsme dali sekci s ceníkem třídu ‚nojs', dle které jsme nastylovali základ, který se bude zobrazovat i v případě nefunkčnosti či nepřítomnosti JS. Tento způsob zobrazení sice nevypadá příliš lákavě, ale poskytuje uživateli veškeré dostupné informace za každých podmínek (viz *Obr. 14*).



Obr. 14: Zobrazení ceníku bez JS

(Zdroj: Vlastní tvorba)

Pro původně zamýšlený efekt nám poté stačily pouhé dva řádky JS upravující třídu elementu a v CSS jsme zvlášť nastylovali tabulky s cenami podle toho, zda jejich rodič má třídu ‚js' nebo ‚nojs' a jelikož jsou tlačítka na stejné úrovni jako tabulky, mohli jsme zprovoznit schovávání a zobrazování tabulek téměř čistě pomocí CSS. V JS jsem pouze přiřadili tlačítkům funkci, která jim přidává nebo odebírá třídu ‚active', díky které jsme specifikoval chování ve stylech (viz *Obr. 16*).

```
$( document ).ready(function() {
    //pokud se správně načte JS, stanou se níže napsané události

    //najde všechny bloky s třídou "nojs", tu odebere a naopak přídá třídu "js"
    $('.nojs').removeClass('nojs').addClass('js');

    //při kliknutí na tlačítka s rodičem #pricelist nejprve najde zrovna aktivní tlačítko,
    //kterému odebere třídu "active" a tu přidělí tlačítku, na které uživatel zrovna kliknul
    $('#pricelist button').on('click', function() {
        $('button.active').removeClass('active');
        $(this).addClass('active');
    });

```

Obr. 15: Funkce pro zprovoznění interaktivního ceníku

(Zdroj: Vlastní tvorba)

Samotné schovávání a zobrazování jsme vyřešili tak, že jsme tabulkám nastavili absolutní pozici, která dovoluje pozicovat element dle libosti, a umístili je mimo dohled uživatele. Pomocí selektoru sourozence ‚~' jsme byli poté schopni jednotlivé tabulky umístit na správné místo právě ve chvíli, kdy jim náležící tlačítko mělo zrovna třídu ‚active' (viz *Obr.16*). Třídu ‚active' jsme ze základu také nastavili prvnímu tlačítku, aby byla při prvním načtení zobrazena aspoň nějaká tabulka. S absolutní pozicí tabulek však přišly i některé problémy týkající se velikosti a uspořádání celého bloku s ceníkem. Museli jsme tedy v tomto případě nastavit pevnou výšku bloku a různých odsazení, aby za všech okolností blok nepůsobil rozbitě, jelikož elementy s absolutní pozicí na rozdíl od jiných neovlivňují velikost jejich rodičovského elementu.

```
.js .soft-cont,
.js .birch-cont,
.js .hard-cont,
.js .other-cont {
    position: absolute;
    width: 95%;
    top: 11.5rem;
    transition: 0.5s ease;
}

.js .soft-cont, .js .hard-cont {
    left: -100%;
    opacity: 0;
}

.js .birch-cont, .js .other-cont {
    left: 100%;
    opacity: 0;
}

.js .soft.active ~ .soft-cont,
.js .hard.active ~ .hard-cont,
.js .birch.active ~ .birch-cont,
.js .other.active ~ .other-cont {
    left: 0;
    right: 0;
    opacity: 1;
}
```

Obr. 16: Schovávání a zobrazování tabulek

(Zdroj: Vlastní tvorba)

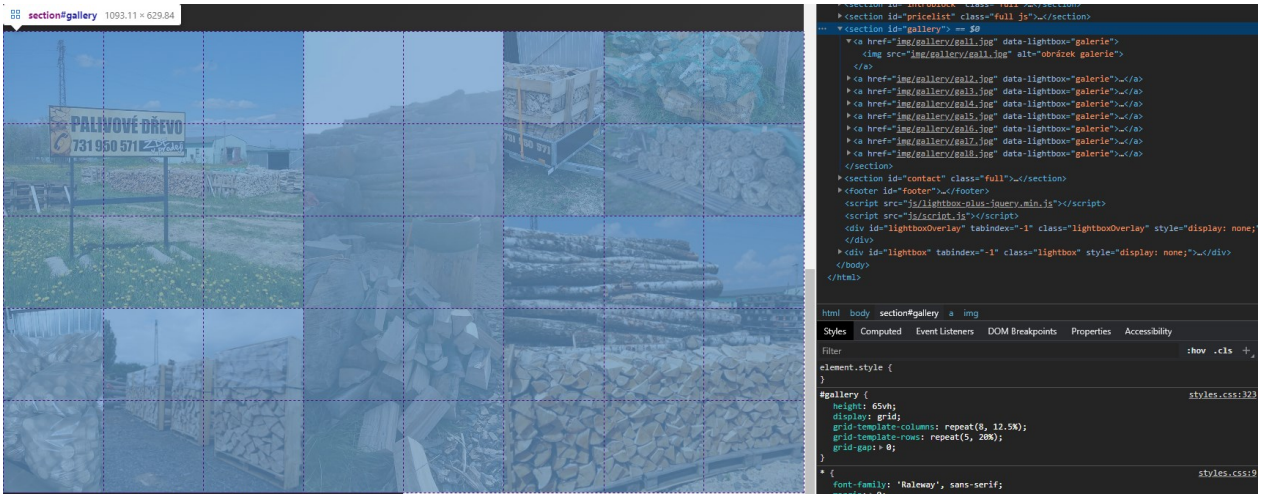
Jelikož tabulky ceníku neobsahují příliš informací, přišlo nám vhodné rozdělit celý blok s ceníkem na poloviny a do druhé půlky vložit blok s doplňujícími informacemi, jak je tomu na původních stránkách. V této chvíli jsme si vzali za vzor frameworky a knihovny a definovali si vlastní třídy určující specifická pravidla, které jsme dále využívali i pro další elementy na stránce, nebo jsme počítali s tím, že je využijeme. Kdykoliv jsme potřebovali udat nějakému elementu například poloviční šířku, nebo zalamování elementů, měli jsem tyto třídy k dispozici a stačilo je přiřadit v HTML (viz *Obr. 17*).

```css
.col25 {
    /* Čtvrtinová šířka */
    width: 25%;
}

.col33 {
    /* Třetinová šířka */
    width: 33%;
}

.col50 {
    /* Poloviční šířka */
    width: 50%;
}

.full {
    /* Plná šířka */
    width: 100%;
}

.wrap {
    /* Zalamování elementů */
    display: flex;
    flex-wrap: wrap;
}

.nowrap {
    /* Žádné zalamování, vše bude na jednom řádku */
    display: flex;
    flex-wrap: nowrap;
```

Obr. 17: Vlastní přednastavené třídy

(Zdroj: Vlastní tvorba)

Další na řadě byla sekce galerie. Chtěli jsme dosáhnout mozaikového efektu a zároveň se vyhnout dalším externím JS souborům, takže jsme se rozhodli použít technologii CSS grid. Předem jsme si zhruba načrtnuli rozpoložení fotek a poté jsme fotku jednu po druhé pozicovali do připravené mřížky. Jedinou nevýhodou je, že tento proces byl poměrně zdlouhavý a zápis může působit zmatečně. Také jsme ve výsledku museli některé fotky vyměňovat, ale výsledek splnil naše očekávání. Při tvorbě nám také velmi pomohly vývojářské nástroje prohlížeče Chrome, který pro CSS Grid zobrazuje mřížku přímo v prohlížeči a v případě chyby nebo ujasnění pozice stačil pouze jeden pohled a věděli jsme v čem chyba spočívá (viz *Obr. 18*).



Obr. 18: Zobrazení mřížky za pomocí vývojářských nástrojů Chrome

(Zdroj: Vlastní tvorba)

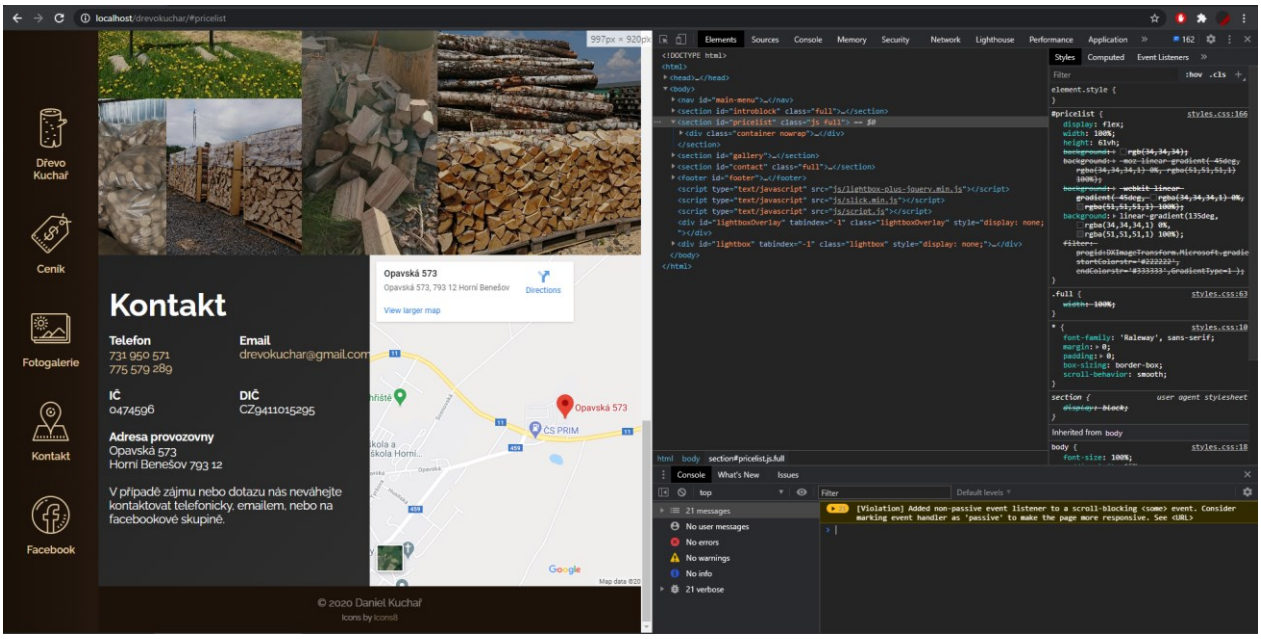
Při tvorbě galerie jsme také využili jedinou externí JS knihovnu a tedy Lightbox, díky kterému stačí každému odkazu na obrázek přidat parametr ‚data-lightbox' a obrázek se následně automaticky při kliknutí zobrazí v připraveném vyskakovacím okně, ve kterém může uživatel překlikávat na jiné obrázky a prohlédnout si celou galerii na jednom místě, namísto aby odkazoval mimo stránky pouze na obrázek.

Posledním složitějším blokem byla sekce s Kontaktem. Zde jsme měli problémy zejména s vizuální reprezentací. Naší vizí bylo, aby se první polovina s informacemi řadila do containeru a byla odsazena jako všechny textové elementy na stránce, zatímco druhá polovina s mapou měla tento container přesahovat mimo až po okraj stránky. Samotnou mapu jsme vytvořili pomocí online nástroje *Google Maps iframe Generator | Add Google Map to Website | Embed Google Maps HTML*[25], do kterého stačí pouze zadat libovolnou adresu a dostaneme tzv. ‚iframe' element, který můžeme vložit do našeho kódu a jednoduše tak získáme interaktivní mapu. Stejně jako ostatní elementy, i ‚iframe' element můžeme libovolně stylovat pomocí CSS, což nám dovolilo nastavit mapě absolutní pozici a rozšířit ji i mimo container, abychom docílili požadovaného efektu rozšíření do konce stránky.

[25] Dostupné z: www.maps.ie/create-google-map/

Responzivita

Po doplnění zbývajících elementů textem jsme považovali stránku za hotovou pro desktopová zařízení. Na řadě tedy bylo začít testovat responzivitu a obohacovat CSS o speciální pravidla pomocí media queries. Pro kontrolu responzivity jsme opět používali vývojářské nástroje v prohlížeči Google Chrome. Hlavní výhodou je to, že když jsou tyto nástroje aktivní, zobrazuje se při změně velikosti přímo v prohlížeči aktuální rozměr okna v pravém horním rohu (viz *Obr. 19*). Díky této funkci můžeme jednoduše určovat, v jakých rozměrech nastávají chyby a použít tyto rozměry pro tvorbu breakpointů. Prvně jsme zjistili, že pro rozlišení 1000–1920 pixelů žádné problémy nenastávají, ale lehce pod 1000 pixelů už bylo viditelné, že některé elementy se začínají překrývat a nejsou správně viditelné. Abychom se vyhnuli vytváření přebytečného množství breakpointů, zmenšovali jsem okno až do chvíle, kdy už bylo třeba schovávat hlavní menu, což ve výsledku připadlo na šířku 800 pixelů. Vyhledali jsem proto všechny chyby v rozmezí 1000 a 800 pixelů a všechny zahrnuli do prvního breakpointu od 1000 pixelů níže. V tomto případě šlo poměrně jednoduché úpravy týkající se hlavně zmenšování textů a roztahování elementů, které původně zabíraly polovinu containeru, na plnou šířku.



Obr. 19: Kontrola rozlišení pomocí vývojářských nástrojů

(Zdroj: Vlastní tvorba)

Největší zádrhel tedy přišel na šířce 800 pixelů, kdy bylo třeba zprovoznit mobilní menu. To se sebou neslo řadu problémů a dodatečných úprav. K menu jsme museli přidat tlačítko sloužící pro jeho otevírání a zavírání. Doufali jsme, že nám funkcionálně postačí otevírání menu pomocí tzv. ‚hover' události, která na počítači reprezentuje najetí myši a na mobilních zařízeních kliknutí, protože pomocí CSS nemůžeme ovlivňovat události kliknutí. Do jisté míry vše pracovalo, jak mělo, ale při simulaci mobilních zařízení (opět pomocí vývojářských nástrojů prohlížeče) jsme zjistili, že tlačítko pro zavírání menu nefunguje tak, jak jsme předpokládali. V tomto případě jsme se bohužel neobešli bez dalších JS funkcí a pro jistotu jsme rozdělili některé dosavadní styly menu do specifických media query. Vše, co bylo ovlivňováno efektem najetí, jsme omezili pouze na větší rozměry, zatímco u menších rozměrů jsme všechny efekty najetí změnili na kliknutí pomocí JS (viz *Obr. 20*).

```
14    //pokud je menu otevřené a klikneme na cokoliv mimo menu, zavře se
15    jQuery(document.body).on("click", ":not(#main-menu, #main-menu *)", function(){
16        $('.open').removeClass('open');
17    });
18
19    //při kliknutí na tlačítko pro otevírání/zavírání menu se přiřadí/odebere třída 'open'
20    $('.menu-button').on('click', function() {
21        ($('#main-menu').toggleClass('open'));
22    });
```

Obr. 20: Ošetření menu na mobilních zařízeních

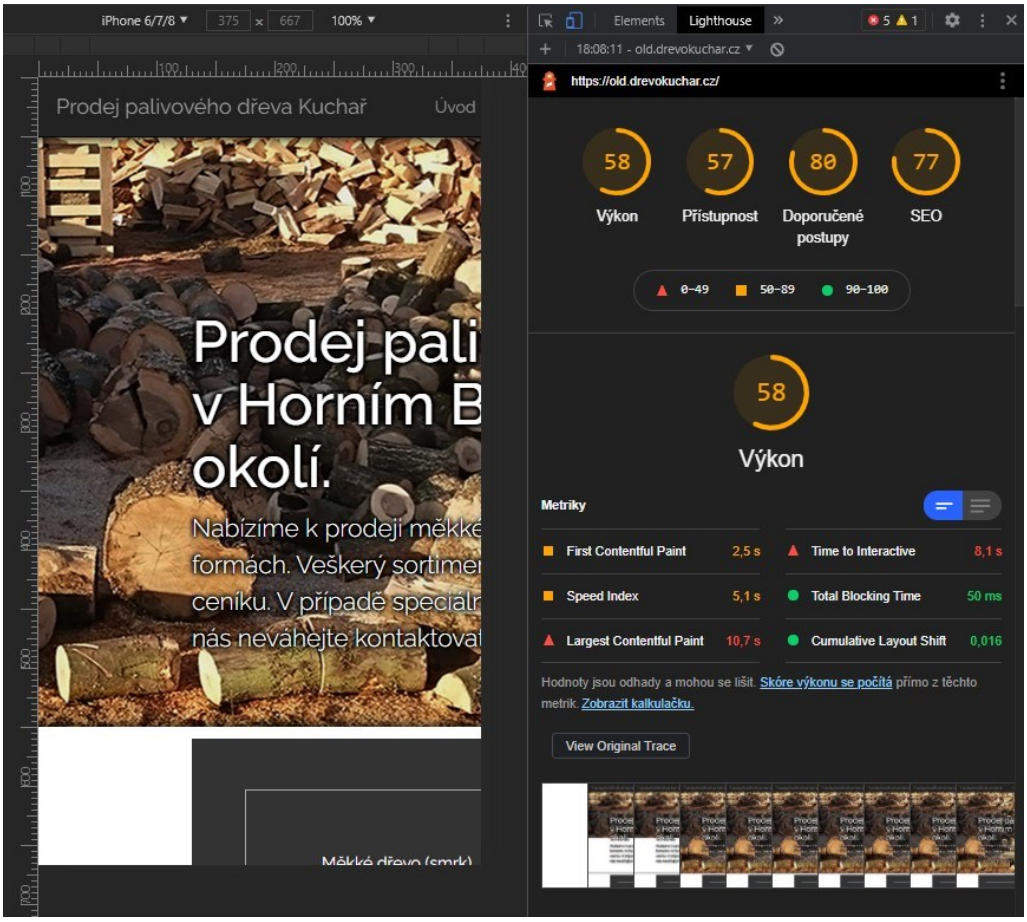
(Zdroj: Vlastní tvorba)

Dále jsme pokračovali již se zaměřením na mobilní zařízení, a to šířky 500 pixelů a méně. Nejsložitější zde byla úprava galerie, kterou jsme museli celou předefinovat fotku po fotce do nové mřížky, která více seděla na mobilní obrazovky. U jiných elementů jsme na problémy nenarazili, protože se většinou opakoval předchozí proces – zmenšování textu a úpravy velikostí bloků, aby nebylo nic schováno nebo se nic nepřekrývalo.

Konečné porovnání a kontrola

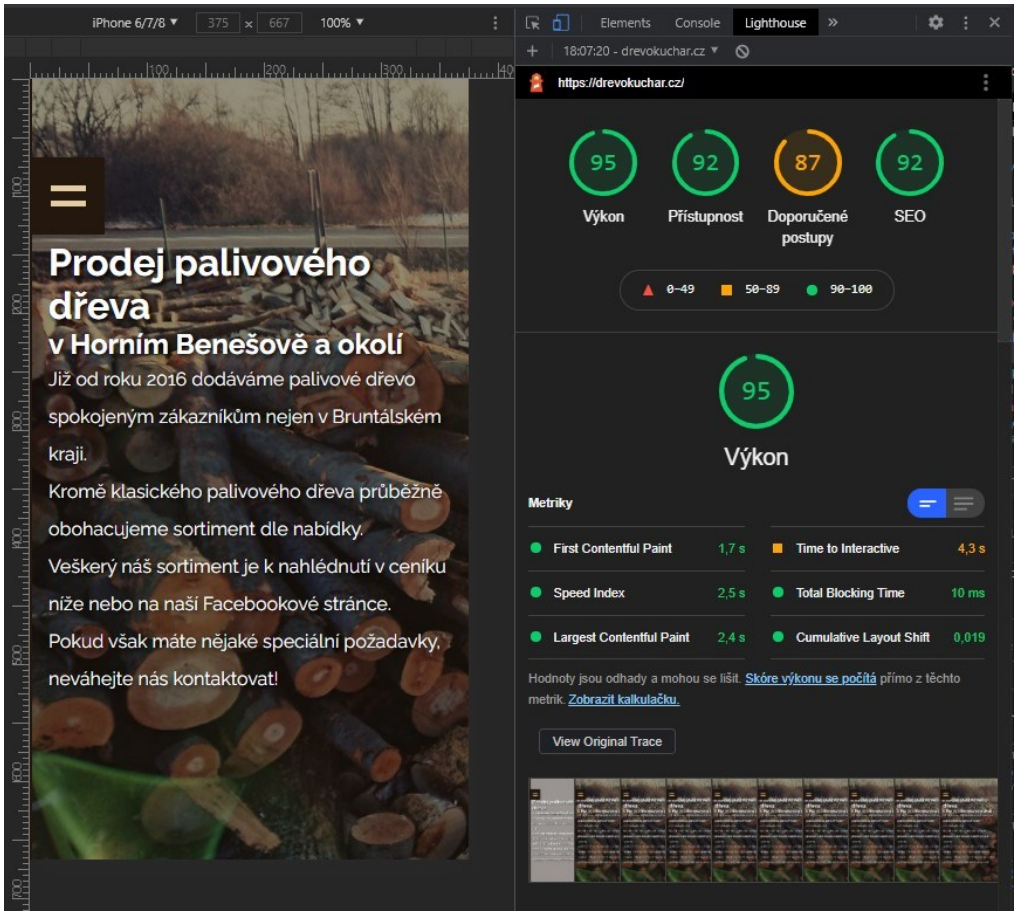
Na závěr jsme nový web porovnali se starou verzí jak na simulovaných zařízeních přes prohlížeč, tak na dostupných mobilních zařízeních, které jsme měli po ruce a rozdíl mezi jednotlivými weby byl více než znatelný. V tomto případě jsou simulace v prohlížeči přesné

a ukazují věruhodné výsledky v porovnání se skutečnými mobilními zařízeními. Rovnou jsme se rozhodli otestovat oba weby pomocí nástroje Lighthouse, který číselně vyhodnotil audit obou webů v mnoha ohledech. Lighthouse je součástí prohlížeče Google Chrome a můžeme si vybrat, zda má hodnotit web v desktopové nebo mobilní verzi. Pro potřeby této práce jsme se rozhodli brát jako hlavní hodnocení právě mobilní verzi, která bývá zpravidla přísnější.



Obr. 21: Vyhodnocení auditu a simulace staré verze webu.

(Zdroj: Vlastní tvorba)



Obr. 22: Vyhodnocení auditu a simulace nové verze webu

(Zdroj: Vlastní tvorba)

Již od pohledu je zřejmé, že web si polepšil nejen po vzhledové stránce, ale také si polepšil po stránkách přístupnosti a SEO.

Závěr

Hlavním cílem práce bylo prakticky implementovat moderní webové technologie a postupy za účelem aktualizace vybrané webové stránky, která pak bude splňovat moderní standardy v oblasti responzivity. Dílčím cílem práce bylo představit moderní webové technologie a přístupy při tvorbě responzivního webdesignu.

Bakalářská práce byla rozdělena na část teoretickou a praktickou.

První zmíněná obsahovala celkem čtyři kapitoly. Ty byly v plné návaznosti na cíle práce a byly zaměřeny na představení moderních webových technologií a přístupů při tvorbě responzivních webových stránek. Při psaní teoretické části jsme narazili na několik problémů. Ty se týkaly právě velkého množství informací a různých postojů v souvislosti se zkoumanou problematikou. Webdesign je neustále se vyvíjející disciplína. Existuje velké množství způsobů jeho realizace, které se odvíjí od schopností a možností autora (programátora). Proto jsme se rozhodli vybrat právě takové, jejichž implementaci zvládne i mírně pokročilý uživatel. Často jsme se také setkávali s informačními zdroji, které se rozcházely v definicích jednotlivých pojmů nebo používaly výrazy, jejichž význam se za poslední roky či měsíce pozměnil a mohly by působit zmatečně.

Obsah praktické části odpovídá páté kapitole této práce. Tato se zabývala implementací vybraných moderních webových technologií představených v teoretické části. Výstup praktické části plně koresponduje s hlavním cílem této práce. V praktické části byly vybrány vhodné technologie a postupy pro dosažení responzivity u konkrétní webové stránky. Postup tohoto zlepšení byl v praktické části obohacen průvodními obrázky, doporučeními a rychlým zhodnocením výsledného rozdílu. Konečná kontrola také ukázala, že aplikované metody splnily svůj účel a podařilo se nám tak přiblížit i problematiku přístupnosti webových stránek, která je s responzivitou úzce spojena a je na ni brán čím dál větší zřetel.

Cíle práce byly splněny.

Domníváme se, že témata zahrnutá v teoretické i praktické části nabízí mnoho prostoru pro jejich rozšíření a další navázání jinými autory. Práce představila obecný pohled na věc a základní stavební kameny webových stránek a mohla by sloužit jako průvodní dokument pro zájemce o tuto problematiku nebo jako materiál pro její budoucí rozšíření.

Seznam použitých zdrojů

ManagementMania, 2015. *Webdesigner* [online]. Dostupné z: https://managementmania.com/cs/webdesigner

CASTIGLIONE, Chris, 2019. *Frontend vs. Backend Developers: What's the Difference?* [online]. Dostupné z: https://learn.onemonth.com/frontend-vs-backend-developers/

PEKARSKY, Max, 2020. *Does your web app need a front-end framework?* [online]. Dostupné z: https://stackoverflow.blog/2020/02/03/is-it-time-for-a-front-end-framework/

RADUCAN, Veronica, 2018. *Website layouts that make a website experience memorable* [online]. Dostupné z: https://colibriwp.com/blog/website-layout-design-ideas/

CASTIGLIONE, Chris, 2020. *Responsive vs. Adaptive vs. Fluid Design* [online]. Dostupné z: https://learn.onemonth.com/responsive-vs-adaptive-vs-fluid-design/

LEPAGE, Pete a ANDREW, Rachel, 2020. *Responsive web design basics* [online]. Dostupné z: https://web.dev/responsive-web-design-basics/

BOSE, Shreya, 2019. *Defining Breakpoints in Responsive Web Design* [online]. Dostupné z: https://www.browserstack.com/guide/responsive-design-breakpoints

XIA, Vincent, 2017. *What is Mobile First Design? Why It's Important & How To Make It?* [online]. Dostupné z: https://medium.com/@Vincentxia77/what-is-mobile-first-design-why-its-important-how-to-make-it-7d3cf2e29d00

BRACEY, Kezz, 2015. *Comprehensive Guide: When to Use Em vs. Rem* [online]. Dostupné z: https://webdesign.tutsplus.com/tutorials/comprehensive-guide-when-to-use-em-vs-rem--cms-23984

Statcounter, 2020. *Web Analytics Made Easy – Statcounter* [online]. Dostupné z: https://statcounter.com/

COYIER, Chris, 2020. *CSS-Tricks* [online]. Dostupné z: https://www.css-tricks.com

EYGI, Cem, 2020. *Media Query CSS Tutorial – Standard Resolutions, CSS Breakpoints, and Target Phone Sizes* [online]. Dostupné z: https://www.freecodecamp.org/news/css-media-queries-breakpoints-media-types-standard-resolutions-and-more/

W3C, 2014. *HTML5 Differences from HTML4* [online]. Dostupné z: https://www.w3.org/TR/html5-diff

Refsnes Data, 2020. *W3Schools Online Web Tutorials* [online]. Dostupné z: https://www.w3schools.com/

W3C, 2017. *Media Queries Level 4* [online]. Dostupné z: https://www.w3.org/TR/mediaqueries-4

KNIGHT, Kayla, 2019. *Fixed vs. Fluid vs. Elastic Layout: What's The Right One For you?* [online]. Dostupné z: https://www.smashingmagazine.com/2009/06/fixed-vs-fluid-vs-elastic-layout-whats-the-right-one-for-you/

The jQuery Foundation, 2020. *jQuery API Documentation* [online]. Dostupné z: https://api.jquery.com/

WOZNIEWICZ, Brandon, 2019. *The Difference Between a Framework and a Library.* [online]. Dostupné z: https://www.freecodecamp.org/news/the-difference-between-a-framework-and-a-library-bd133054023f/

UX Alpaca, 2016. *What is the difference between fixed, fluid, adaptive and responsive layouts and why should I care?* [online]. Dostupné z: https://medium.com/@space.alpaca/so-what-exactly-is-the-difference-between-fixed-fluid-adaptive-and-responsive-layouts-and-why-3773272d8481

COCHRAN, David. *Twitter Bootstrap Web Development How-To*. Packt Publishing Ltd, UK, 2012. ISBN 978-1849518826

Twitter, 2020. *Bootstrap · The most popular HTML, CSS, and JS library in the world.* [online]. Dostupné z: https://getbootstrap.com/

ZURB, 2020. *The most responsive front-end framework in the world.* [online]. Dostupné z: https://get.foundation

GENGE, David, 2018. *Foundation vs. Bootstrap: Which front end framework to use?* [online]. Dostupné z: https://medium.com/@davegenge/bootstrap-vs-foundation-which-front-end-framework-to-use-e85319258b88

OpenJS Foundation, 2020. *AMP - a web component framework to easily create user-first web experiences - amp.dev* [online]. Dostupné z: https://amp.dev

SHEFFIELD, Matthew, 2017. *Russian hackers exploited a Google flaw to hack journalists* [online]. Dostupné z: https://www.salon.com/2017/09/24/russian-hackers-exploited-a-google-flaw-and-google-wont-fix-it/

GILBERTSON, Scott, 2017. *Kill Google AMP before it killst he web* [online]. Dostupné z: https://www.theregister.com/2017/05/19/open_source_insider_google_amp_bad_bad_bad/

Codecademy, 2020. *What is an IDE* [online]. Dostupné z: https://www.codecademy.com/articles/what-is-an-ide

Microsoft, 2020. *Visual Studio Code - Code Editing. Refined* [online]. Dostupné z: https://code.visualstudio.com

CARBONNELLE, Pierre, 2020. *Top ID index* [online]. Dostupné z: https://pypl.github.io/IDE.html

ESCALONA, Armela, 2017. *The Pros and Cons of Using Website Builders* [online]. Dostupné z: https://www.templatemonster.com/blog/pros-cons-using-website-builders/

L., Sergey, 2018. *Web Design Process: 7 Steps on How We Create Websites Appearance At Cleveroad* [online]. Dostupné z: https://www.cleveroad.com/blog/web-design-process

GRAHAM, Geoff, 2015. *The Difference Between Responsive and Adaptive Design* [online]. Dostupné z: https://css-tricks.com/the-difference-between-responsive-and-adaptive-design/

WROBLEWSKI, Luke, 2009. *Mobile First* [online]. Dostupné z: https://www.lukew.com/ff/entry.asp?933

EVERTS, Tammy, 2016. *Mobile Load Time and User Abandonment* [online]. Dostupné z: https://developer.akamai.com/blog/2016/09/14/mobile-load-time-user-abandonment

WROBLEWSKI, Luke, 2014. An Event Apart: *Mobile First Responsive Design* [online]. Dostupné z: https://www.lukew.com/ff/entry.asp?1873

KAEMPF, Daniel, 2018. *Choose your strategy: Mobile-First Web Design vs. Responsive Web Design* [online]. Dostupné z: https://darwindigital.com/mobile-first-versus-responsive-web-design/

MEYER, Eric A. a MEYER, Kathryn S., 2020. CSS Tools: Reset CSS [online]. Dostupné z: https://meyerweb.com/eric/tools/css/reset/

Voormedia, 2020. *TinyPNG – Compress PNG images while preserving transparency.* [online]. Dostupné z: https://tinypng.com/

GALLAGHER, Nicolas, 2012. *About normalize.css* [online]. Dostupné z: https://necolas.github.io/normalize.css/

COYIER, Chris, 2017. *Reboot, Resets, and Reasoning* [online]. Dostupné z: https://css-tricks.com/reboot-resets-reasoning/

DEVERIA, Alexis, 2020. *Can I use… Support tables for HTML5, CSS3, etc* [online] Dostupné z: https://caniuse.com/

Digital Education Strategies, The Chang School. *Web Accessibility for Developers* [online]. The Chang School, Ryerson University, 2019 [cit. 15.7.2020]. Dostupné z: https://pressbooks.library.ryerson.ca/wafd/chapter/types-of-disabilities-and-barriers/

BOS, Bert, ÇELIK, Tantek, HICKSON, Ian a LIE, Håkon Wium. *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification* [online]: Dostupné z: https://www.w3.org/TR/CSS21/media.html%23media-types

Seznam použitých obrázků

Anotace

Jméno a příjmení:	Daniel Kuchař
Katedra:	Katedra technické a informační výchovy
Vedoucí práce:	Mgr. Tomáš Dragon
Rok obhajoby:	2021

Název práce:	Tvorba webových stránek pro různá zobrazovací zařízení
Název v angličtině:	Webdesign for various display devices
Anotace práce:	Bakalářská práce se zaměřuje na postupy a dostupné technologie při tvorbě responzivních webových stránek. Ty jsou následně v teoretické části představeny a popsány. V praktické části jsou teoretické informace aplikovány na konkrétní webovou stránku za účelem zlepšení responzivity a celkového hodnocení webu.
Klíčová slova:	CSS, JS, media query, responzivita, HTML, webová stránka, framework, adaptibilita, přístupnost webu
Anotace v angličtině:	This bachelor's thesis focuses on modern technologies and practices used in web development. These are introduced and described in the theoretical part, whereas practical part consists of applying mentioned technologies and practices on a specific live web in order to improve its responsivity and overall performance.

Klíčová slova v angličtině:	CSS, JS, media query, responsivity, HTML, web page, framework, adaptibility, web accessibility
Přílohy vázané v práci:	přílohy: X
Rozsah práce:	52 stran
Jazyk práce:	čeština