Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Diplomová práce

Postupy tvorby responsivního designu

Bc. Lukáš Modrý

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Lukáš Modrý

Informatika

Název práce

Postupy tvorby responsivního designu

Název anglicky

Procedures of responsive design creating

Cíle práce

Diplomová práce je tematicky zaměřena na problematiku responsivního designu. Hlavním cílem práce je komparace možností tvorby responsivního designu s následným pilotním ověřením na reálných příkladech.
Dílčí cíle práce jsou:
- sumarizace vývoje CSS,
- komparace CSS frameworků.

Metodika

Metodika řešené problematiky diplomové práce je založena na studiu a analýze odborných informačních zdrojů. Vlastní práce spočívá v aplikaci jednotlivých možností tvorby responsivního designu na ukázkových příkladech a jejich následné testování na různých zařízeních. Dále zhodnocení a doporučení vhodnosti jednotlivých možností v konkrétních případech. Na základě syntézy teoretických poznatků a výsledků praktické části budou formulovány závěry diplomové práce.

Doporučený rozsah práce

60 – 80 stran textu.

Klíčová slova

Responsivní design, CSS, framework, Media Queries, přístupnost, webdesign, mobilní web

---

Doporučené zdroje informací

Craig Sharkie, Andrew Fisher. Responzivní webdesign okamžitě. 1. Vydání. Brno: Computer Press, 2015, 144 s. ISBN 978-80-251-4384-1.

Echer, Clint. Profesionální webdesign Techniky a vzorová řešení. 1. vydání. Brno: CP Books, a.s., 2005. 421s. ISBN 80-251-0547-4.

Ethan Marcotte. Responsive Web Design, A Book Apart, 2011, 150 s. ISBN: 978-098442577.

Media Queries. W3C. [Online] http://www.w3.org/TR/css3-mediaqueries

Tim Kadlec. Responzivní design – profesionálně. Zoner Press, 2014, 248 s. ISBN: 978-80-7413-280-3

Webdesigner Depot [Online] http://www.webdesignerdepot.com

Zdroják.cz [Online] http://www.zdrojak.cz

Předběžný termín obhajoby

2015/16 LS – PEF

Vedoucí práce

Ing. Pavel Šimek, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 28. 10. 2015

Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 11. 11. 2015

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 28. 02. 2016

Prohlašuji, že svou diplomovou práci "*Postupy tvorby responsivního designu*" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 31.3.2016 .....

Vlastnoruční podpis

Touto cestou bych rád vyjádřil velké díky panu Ing. Pavlu Šimkovi, Ph.D. za jeho odborné rady, trpělivost a ochotu při vedení diplomové práce. Dále bych chtěl poděkovat rodině a blízkým za podporu během celého studia.

.....

Vlastnoruční podpis

Postupy tvorby responsivního designu

Diplomová práce se zaměřuje na problematiku tvorby responsivního designu a komparaci jednotlivých postupů pro dosažení přizpůsobitelných webových stránek.

V teoretické části práce byla probrána většina dostupných možností pro tvorbu responsivního designu a vysvětleno jejich obecné použití.

Praktická část se zaměřuje na použití jednotlivých postupů na konkrétní webové prezentaci. V práci byly vytvořeny tři podobné webové stránky, kde byly využity jednotlivé postupy tvorby responsivního designu bez použití frontendových frameworků a následně s jejich využitím. Na základě jejich testování bylo poté vzneseno doporučení k jejich využití v praxi.

Součástí práce je rovněž shrnutí vývoje kaskádových stylů se zaměřením na responsivní design a dále komparace dvou zvolených frontendových frameworků.

Klíčová slova

Responsivní design, CSS, framework, Media Queries, přístupnost, webdesign, mobilní web

Procedures of responsive design creating

This Master‘s thesis focuses on the issue of responsive design creation and comparison of the particular patterns aiming to customizable website. The theoretical part discusses the majority of available possibilities for achieving a responsive design and their general purpose. The practical part covers the use of the particular patterns on the specific web presentation. The thesis includes three similar websites using the responsive design particular patterns without frontend frameworks and, subsequently, their utilization. Their testing served as a means for recommendation for the practice. Another part of the thesis is the summarization of cascading style sheets development with the focus on the responsive design and comparison of two frontend frameworks selected by the author.

Keywords

Responsive design, CSS, framework, Media Queries, accessibility, webdesign, mobile web

Obsah

1 Úvod

Internet je jediným místem na světě, kde je v jednom okamžiku možné oslovit miliony lidí napříč celou naší planetou. Stal se hlavním komunikačním kanálem mnoha firem a také místem, kde se každodenně setkávají lidé jak se svou prací, tak i přáteli a rodinou. Značná část dnešní populace si nedovede představit život bez přístupu k internetu, což zapříčinilo i značný rozvoj technologií s možností připojení k síti.

S rozvojem technologií narůstají také nároky uživatelů na prohlížení webu napříč jednotlivými zařízeními a na požitky, které jim webové prezentace poskytují. Starší webové stránky, vytvořené zastaralými postupy, začínají být na moderních zařízeních značně neovladatelné. To především z důvodu stále se zmenšujících velikostí obrazovek s naopak neustále se zvyšujícím rozlišením zobrazení. Zastaralé metody pro tvorbu webových stránek přestaly být dostačující a bylo třeba je nahradit novými možnostmi pro vytváření internetového obsahu, které uspokojí potřeby uživatelů.

Responsivní web je moderní pojetí pro tvorbu webových stránek, jež se automaticky přizpůsobují možnostem zařízení, na kterém jsou zobrazovány. Internet je však velmi složité a stále se vyvíjející prostředí, díky čemuž vzniklo množství různých postupů a možností pro tvorbu těchto stránek. Možnosti tvorby responsivního webu se snaží doplňovat i konsorcium W3, které do specifikace HTML a CSS implementuje nové prvky, jež vývojářům tvorbu responsivních stránek usnadní. Problémem je podpora jednotlivých alternativ ve webových prohlížečích jak pro desktopy, tak pro mobilní zařízení.

Webové stránky se díky rozšíření internetu mezi širokou veřejnost staly také jedním z hlavních míst obchodu a reklamy. Dnešní rychlá doba však často neposkytuje uživatelům klid a pohodlí pro realizaci konverzí na webových stránkách a je tak nutné jim tuto činnost usnadnit. Při tvorbě webové stránky tak není hlavním požadavkem přizpůsobení stránky jednotlivým zařízením, ale jednotlivým uživatelům a situacím, ve kterých budou tito uživatelé k webu přistupovat.

2 Cíl práce a metodika

2.1 Cíl práce

Diplomová práce je tematicky zaměřena na problematiku responsivního designu. Hlavním cílem práce je komparace možností tvorby responsivního designu s následným pilotním ověřením na reálných příkladech.

Dílčí cíle práce jsou:

- sumarizace vývoje CSS,

- komparace CSS frameworků.

2.2 Metodika práce

Metodika řešené problematiky diplomové práce je založena na studiu a analýze odborných informačních zdrojů.

Vlastní práce spočívá v aplikaci jednotlivých možností tvorby responsivního designu na ukázkových příkladech a jejich následné testování na různých zařízeních. Dále zhodnocení a doporučení vhodnosti jednotlivých možností v konkrétních případech. Na základě syntézy teoretických poznatků a výsledků praktické části budou posléze formulovány závěry diplomové práce.

Samotná práce se dělí do 3 základních kapitol:

- 3. Teoretická východiska práce
- 4. Praktická část
- 5. Zhodnocení výsledků a doporučení

Kapitola *„Teoretická východiska práce“* se zabývá problematikou tvorby responsivního designu a většiny základních metod pro jeho tvorbu. V kapitole jsou vysvětleny základní pojmy spojené s tímto tématem. Dále jsou zde podrobně popsány jednotlivé možnosti tvorby responsivního designu.

„Praktická část“ práce je zaměřena na samotnou tvorbu responsivní webové prezentace bez použití dostupných frameworků. V kapitole jsou popsány jednotlivé kroky tvorby konkrétní webové stránky. Některé dílčí prvky jsou poté vytvořeny také za pomocí komponent frameworků Bootstrap 3.3.6 a Foundation 6 a následně je zhodnocena jejich použitelnost a přizpůsobitelnost, což splní dílčí cíle práce.

Na základě těchto přístupů bude na konci každé podkapitoly praktické části vyhodnocena nejvhodnější alternativa daného responsivního prvku pro použití v praxi. Samotné porovnání jednotlivých přístupů bude vyhodnoceno na základě praktického příkladu a jeho proveditelnosti a podpory jednotlivých vlastností v prohlížečích. Dále bude testováno chování jednotlivých možností napříč prohlížeči a zařízeními. Samotné testování bude prováděno s použitím většiny v době psaní této práce dostupných a obvykle používaných webových prohlížečů.

V kapitole *„Výsledky a jejich zhodnocení“* budou vyhodnoceny výstupy z praktické části jako celku. Dále bude zhodnocena možnost využívání frontendových frameworků pro vytváření responsivních webových stránek.

3 Teoretická východiska práce

3.1 Webdesign

Pojem webdesign má své základy už na počátku veřejného internetu. Díky množství nových technologií se však stal velmi širokým a obsáhlým tématem. Je to mezioborová disciplína, která se skládá z poznatků komunikace, psychologie, designu, interakce, marketingu, tvorby brandingu, ale také například copywritingu.

Jan Řezáč ve své knize *Web ostrý jako břitva* (Řezáč, 2014) napsal:

„Cílem dobrého webdesignera je vytvořit web, který bude fungovat – tedy plnit svůj účel pro byznys klienta a naplňovat potřeby návštěvníků webu.“

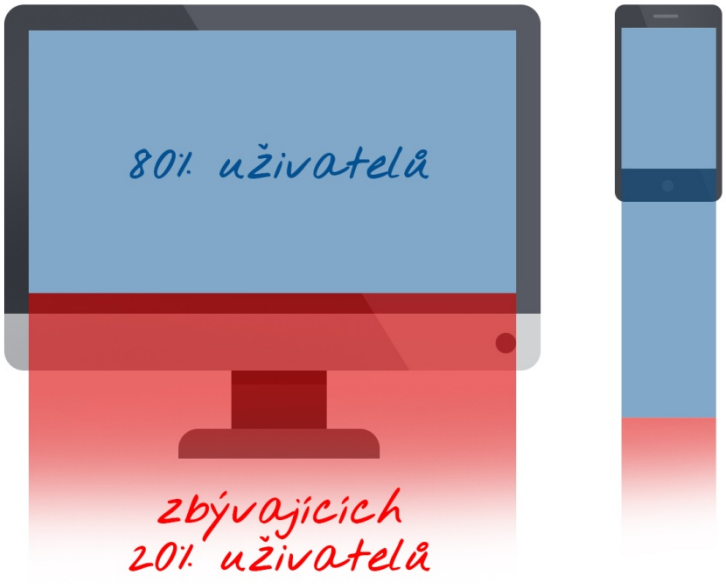
To v praxi znamená, že dobrý webdesigner musí nejen ovládat grafické aplikace pro tvorbu návrhu, ale také pochopit pointu podnikání svého klienta, jeho zákazníky, myšlení cílových uživatelů a mnoho dalšího. Webdesign je průnik vizuální komunikace, obsahové strategie a interakce mezi uživatelem a webem, jež jsou obohaceny vhodnými marketingovými aktivitami. Pointou webdesignu je sjednocení plnohodnotného obsahu s kvalitní grafickou podobou pro dosažení maximálního efektu na uživatele, kteří k webu budou přistupovat. (Řezáč, 2012)

3.2 Princip a požadavky webu

Webové stránky jsou velmi mocným nástrojem reklamy, prodeje a komunikace. (Depot, 2010-2016) Je to nejsnazší způsob, jak oslovit širokou veřejnost. Téměř každá společnost v dnešní době má svou vlastní webovou prezentaci. Přesto vytvoření kvalitních stránek není vůbec jednoduché, a to i v případě, že se sejde schopný designér se schopným vývojářem.

Základním problémem, jenž je nutný řešit při navrhování webových prezentací, je chování uživatele, který ke stránkám přistupuje. Ten zpravidla čte stránky na svém monitoru či displeji standardně zleva doprava, odshora dolů a jednotlivým zobrazeným prvkům přiděluje určitou úroveň svého zájmu. Obsah, který je tedy nejdůležitější, by měl být umístěn co nejvýše na stránce. Studie Jaan-Matti Lillevälji na základě heatmap dokázala, že uživatelé při scrollování stránkou postupně o obsah ztrácejí zájem. Říká, že 80% uživatelů věnuje největší pozornost informacím, které jsou zobrazeny ihned po

načtení stránky. (Lillevälja, 2012) Na následujícím obrázku je znázorněn obsah, který uživatele nejčastěji zajímá a kam až se může posunout při změně rozlišení.



Obrázek 1 - Zájem uživatelů o obsah

S tím vyvstává další problém, spojený s rozlišením obrazovek, na kterých uživatelé stránky zobrazují. Opticky atraktivní a přesto obsahově nezajímavá hlavička může na jednom zařízení zabírat čtvrtinu zorného pole, zatímco na jiném zabere všechen dostupný prostor. Je dokázáno, že mnoho uživatelů už dále neposouvají web dolů a tudíž se k hlavnímu obsahu nedostane. (Dobrý web, 2009)

Nejde však jen o obsah, který je jak pro majitele webové prezentace obchodně důležitý a pro jeho uživatele zajímavý, ale také o celkový pocit z webu. Pokud bude mít návštěvník webové stránky problém s orientací či ovládáním, ztratí o obsah zájem. Tradiční požadavky na web jsou:

- Validní a sémantický zdrojový kód
- Přehlednost
- Dobrá čitelnost
- Web graficky sloužící informační hodnotě (Dobrý web, 2009)

3.3 Standardní rozložení webu

Principy tvorby webových prezentací jdou stále kupředu a požadavky na jejich obsah se neustále rozvíjejí. Přesto však některé standardní prvky webových stránek zůstávají po celou dobu vývoje stejné a jejich umístění je vhodné při navrhování uživatelského prostředí zachovat. Je však potřeba si uvědomit, že množství jednotlivých typů webů je dneska nepřeberné a obsah se všude může podstatně lišit. Základní druhy webových stránek mohou být rozděleny do následujících dvou skupin:

- Webové stránky s klasickým rozložením obsahu
- Stránky se scrollovatelným obsahem, tzv. One page stránky (občas Single page)

Obsahová podoba první a druhé skupiny se v praxi vesměs moc neliší. Rozdíl je pouze ve stylu, jakým jsou informace uživateli předávány.

3.3.1 Klasické rozložení obsahu

Na stránkách s klasickým rozložením obsahu se uživatel proklikává jednotlivými podstránkami webu, aby získal požadovaný obsah. Web je tedy složen z množství jedinečných podstránek vzájemně propojených hypertextovými odkazy. S tímto typem se uživatelé setkávají nejčastěji, jsou tak vytvářeny většinou složitější webové prezentace s větším množstvím obsahu, jako jsou stránky velkých společností, sociální sítě, elektronické obchody apod.

3.3.2 One page design

Oproti tomu One page prezentace poskytují veškeré informace na jedné stránce a uživatel se za účelem posunu obsahu pohybuje především kolečkem myši, případně jiným podobným způsobem. One page design je obvykle navrhován pro vertikální posun, v posledních letech se však často objevují i stránky s posunem horizontálním, či jiným. Tento způsob zobrazování informací je vhodný především pro image stránky společností, které se snaží zaujmout jedinečnou podobou jejich prezentace. Stránky bývají obsahově méně složité, na hlavní straně jsou zobrazeny veškeré podstatné informace a formuláře. Jsou také snadno ovladatelné prostřednictvím dotykových obrazovek. (Chapman, 2010)

Specifickou skupinou takových webů jsou stránky postavené na paralaxovém efektu. Ten umožňuje poskytovat uživateli různý obsah v závislosti na tom, v jaké části webové prezentace se uživatel nachází.

Díky nejmodernějším technologiím se v posledních letech tyto 2 kategorie webů spojují dohromady, přičemž typická One page stránka funguje jako rozcestník k ostatním částem webu, ke kterým se uživatel dostává prostřednictvím hypertextových odkazů. Je však nutné dbát zvýšené pozornosti na hardwarovou náročnost takové stránky. Množství efektů a funkcí je realizováno prostřednictvím skriptovacího jazyka JavaScript a jeho frameworků, nejčastěji JQuery. Ty zvyšují náročnost stránek k jejich načtení a ovládání. (Love, 2014)

3.3.3 Tradiční prvky webů

Přestože je web různorodé prostředí, většina webových stránek má velmi podobný základní obsah.

Téměř na všech webových stránkách je k nalezení logotyp společnosti, které web patří. Dále se zde vyskytuje hlavní navigační panel s odkazy na jednotlivé části webu. Formu a podobu navigačního panelu je nutné důkladně promyslet a vytvořit tak přehledný rozcestník po celých stránkách. Jednotlivé položky by měly být jednoznačně pojmenovány, aby jasně naznačily, kam vlastně povedou. Často vzniká problém s velikostí hlavního panelu, který se nevejde na menší rozlišení obrazovek. Jedním z možných řešení je přeskládání odkazů pod sebe. Dalším způsobem je jeho nahrazení za tlačítko, které se po kliknutí menu zobrazí.

Zažitým prvkem na webových prezentacích je image hlavička, díky které se autoři stránek snaží zaujmout návštěvníky. Často se zde nachází bannery, které se snaží upoutat na aktuální nabídku produktů, akční nabídky, slogany společnosti, apod. Při používání těchto bannerů, jež jsou obvykle zpracovány jako pravidelně se měnící image slider, je však nutno myslet na tzv. *bannerovou slepotu*. Mnoho uživatelů díky všudypřítomné reklamě tyto bannery začíná přecházet a informace v něm obsažené pro ně ztrácejí význam.

Samozřejmostí je hlavní obsahová část webu, kde se nacházejí veškeré podstatné informace, produkty, obsahy jednotlivých stránek a podobně. Díky novým možnostem, které přinesly kaskádové styly třetí generace a HTML5, se na stránkách společností objevuje i značné množství různé infografiky s efekty. Pro zlepšení marketingové komunikace mezi společností a zákazníkem je zvykem na web umisťovat různé kontaktní

formuláře, okna s real-time chatem, aj. (Lillevälja, 2012) V této části se často objevují také různé reklamní bannery, rozcestníkové boxy, apod.

Dalším tradičním prvkem je patička webu se jmény autorů, či produkční společnosti, autorskými právy a další. Není neobvyklý ani výskyt značného množství dalších informací a prvků, jako jsou kontakty, kontaktní formulář, či odkazy na sociální sítě.

U složitějších webových stránek a e-shopů je zvykem do stránek umisťovat různé vyhledávací formuláře, rozcestníkové bannery a ikony, postranní panely s rozšířenou nabídkou menu, či jiné ovládací prvky. U stránek, které pracují s uživatelem, je také nutné do hlavního obsahu stránek umístit další součásti, jako je přihlašovací formulář či nákupní košík. Pro účely vylepšení jak SEO optimalizace, tak i uživatelského rozhraní se často na web umisťuje tzv. drobečková navigace, která pomáhá uživateli zorientovat se o tom, kde se právě nachází. (Gasston, 2015)

3.4 Pravidla tvorby přístupného webu

Obsah webové stránky by měl být dostupný a čitelný, a to na všech zařízeních při všech typech rozlišení obrazovek a odkudkoliv, za jakýchkoliv podmínek. Nejjednodušší definice přístupného webu zní: *„Web je přístupný všude a na všem.“*. Obecnou zásadou je, že dobrý web by měl být přístupný všem skupinám návštěvníků bez rozdílu jejich možností. Tím je myšleno, že webová prezentace by měla být přístupná jak zdravým uživatelům, tak uživatelům s jakýmkoliv zdravotním hendikepem. (Dobrý web, 2009)

Přestože pravidla přístupnosti nejsou předmětem této diplomové práce, je nutné se o nich zmínit především kvůli široké škále různých zařízení pro zobrazování webů, kam spadají i různé čtečky pro nevidomé.

Existuje několik dokumentů, které definují pravidla přístupného webu. Jeden z nejpoužívanějších na území ČR má svůj původ v zákonu, konkrétně v č. 365/2000 Sb. o informačních systémech veřejné správy. (ČR, 2000 - 2009) Vůbec prvním dokumentem na toto téma byl *Web Content Accessibility Guideline*, vydaný v roce 2008. Ten vydalo samotné konsorcium W3, respektive skupina *Web Accessibility Initiative*, která pod konsorcium W3 spadá. Dnes je k dispozici ve verzi 2.0. (W3C, 2008)

Verze WCAG2.0 se snaží pojmout technologie pro tvorbu webu tak, aby pokryla veškeré možné situace, které při prohlížení webu mohou nastat. Samotná dokumentace se dělí na 4 základní části:

- Vnímatelnost
- Ovladatelnost
- Pochopitelnost
- Robustnost (Dobrý web, 2009)

V této části budou popsána jen ta pravidla, která jsou spojena se zobrazením obsahu na různých zařízeních a základní ovladatelností webu napříč zařízeními, typy uživatelů a způsobem čtení obsahu. Tedy konkrétně části: *vnímatelnost, ovladatelnost*.

3.4.1 Vnímatelnost

Jednou ze základních podmínek přístupného webu je, že každý netextový prvek, který nese obsahově významné sdělení, či je klíčový pro ovládání webové stránky, má svou netextovou alternativu. Zrakově postižení uživatelé často využívají různá čtecí zařízení, která nejsou schopná číst obrázky v jejich základní podobě. Při přístupu k webu se určité procento uživatelů setkává s problémem, že není schopno vzhledem ke svému hendikepu správně přečíst a pochopit obsah stránky. Z tohoto důvodu je nutné každému významnému obrázku nastavit jeho textovou alternativu atributem *alt*. Pro obsáhlejší popisy se autor webu může spokojit s dalším atributem pro delší popis obrázky, tedy *longdesc*. Naopak obrázky, které nejsou obsahově významné a slouží pouze jako dekorace webových stránek, mají svůj atribut *alt* prázdný.

```
<img src="obrazek.jpg" alt="krátký popis" longdesc="dlouhý popis">
```

Správný popis obrázků však není jediným problémem při správném popisu netextových prvků. Často je nutné správně identifikovat i formulářové prvky, jako jsou pole pro vkládání textů, nebo tlačítka pro odeslání formuláře. Dále je nutné správně popsat atributem *title* i velmi oblíbené obrázkové odkazy. (Špinar, 2000) Tímto problémem se více zabývá technická specifikace WAI-ARIA (*Web accessible initiative – Accessible Rich Internet Applications)* publikovaná konsorciem W3. Ta rozšiřuje základní specifikaci HTML o nové možnosti pro zlepšení přístupnosti webu. Veškerá zařízení čtou webovou prezentaci jakožto sérii po sobě jdoucích popisků obsahových informací, u kterých nedovedou dále předvídat jejich interakci nebo význam samotného obsahu. Specifikace

přináší řadu různých atributů, které tyto stavy popisují. Tím hlavním je atribut *role*, jenž popisuje obsah. Jednotlivé hodnoty tohoto atributu jsou:

- `application` – pro označení interaktivní aplikace
- `banner` – pro popis záhlaví a obecných informací o stránce (například logo, či image slider)
- `complementary` – pro obsah, související s hlavním obsahem
- `contentinfo` – pro zobrazení informací o obsahu
- `form` – pro popis formuláře
- `main` – pro hlavní obsah webu
- `navigation` – pro hlavní navigační menu
- `search` – pro pole s vyhledáváním (Gasston, 2015)

S dostupností informací je spojeno i využívání množství různých skriptů, appletů, multimediálních prvků, či jiných doplňkových služeb na straně uživatele. Často se s nimi uživatel setkává v podobě různých pop-up oken, záložek, či flash animací apod. Informace jimi zobrazované musí být dostupné i v momentě, že uživatel má konkrétní jazyky a pluginy (například JavaScript) zakázány. To v praxi znamená, že by samotná webová stránka měla obsahovat veškeré informace i bez nutnosti využívání dalších přídavných modulů. WCAG2.0 dále definuje pravidla pro práci s multimediálními prvky. Zde je řečeno, že všechny předtočené audio a video stopy by měly mít svůj textový přepis, synchronní titulky a popis samotného multimediálního obsahu. Dokumentace obsahuje i pravidlo pro zvukové stopy, hrající déle jak 3 vteřiny. Ty by měly být vybaveny tlačítkem pro zastavení, či ztlumení. (Malánek, 2010)

Obsah webu by měl být vytvořen tak, aby se dal zobrazit více způsoby, aniž by došlo ke ztrátě nosných informací. To znamená používat sémanticky významné elementy jazyka HTML správným způsobem. Tedy značky `<h1>` pro nadpisy, `<ul><li><ol>` pro seznamy, `<p>` pro odstavce textu, tabulky pro data, a tak dále. Dále tyto prvky svazovat s jinými prvky, které mají význam v kontextu, tedy například používat záhlaví u tabulek. Vedle toho je nutné dbát i na logické pořadí elementů stránky a samotného obsahu. (Malánek, 2010)

Při tvorbě stránek je často mnoho informací sdělováno prostřednictvím barev. To se týká například ikon skladových zásob u produktů nabízených na elektronických

obchodech, barevně odlišených odkazů v textu, nebo třeba povinných polí ve formulářích. Tyto informace však musí být uživatelům, především těm se zrakovým postižením, poskytnuty i bez nutnosti znalosti rozlišování barevnosti. V dnešní době se to však týká i uživatelů nejmodernější techniky. Například smartphony značky Samsung s operačním systémem Android využívají pro úsporu baterie černobílý barevný model. S barevností jsou však spojeny i barvy pozadí a popředí. Je nutné si uvědomit, že různá zařízení disponují různě kvalitními obrazovkami. Zatímco na drahém IPS monitoru může web vypadat čitelně a použitelně, na levném smartphonu s horším zobrazováním barev už tomu tak být nemusí. Je tedy potřeba dodržovat dostatečný kontrast mezi prvky, které se překrývají. Tento kontrast by měl být v poměru minimálně 3:1. (Špinar, a další, 2006)

Dalším pravidlem je používání velikosti písma, které by nemělo být definováno absolutními jednotkami, jako jsou pixely, milimetry, či palce. Vhodné je tedy používat procentuální jednotky a jednotky *em*, které se odvíjí od normální velikosti. Stránka by pak měla být čitelná i při dvojnásobném zvětšení textu. (Malánek, 2010)

3.4.2 Ovladatelnost

Základním požadavkem je, aby veškeré formuláře, navigace a jiné ovládací prvky byly snadno ovladatelné. To v praxi znamená, že veškeré klíčové a ovládací prvky jsou přístupné z klávesnice. Při procházení stránek prostřednictvím klávesnice by neměla vzniknout situace, kdy se kurzor aktuální pozice zasekne na nějakém prvku a nepůjde se dostat dále. Používání klávesových zkratek by nemělo kolidovat s žádnými standardně používanými klávesovými zkratkami napříč operačními systémy.

Vedle samotné ovladatelnosti je nutné dbát i na přístupnost různého obsahu, který je svázán s nějakým efektem. Například velmi často používané informační slidery by měly být nastaveny na dostatečný časový interval změny, aby si uživatel pohodlně stihl přečíst obsah. Případně by slider měl být vybaven tlačítkem pro stopnutí efektu.

S časovým intervalem je spojeno i odhlášení uživatele. Tato funkce je často využívána například u internetového bankovnictví a jiných služeb s požadavkem na zvýšené zabezpečení. WCAG2.0 říká, že při odhlášení uživatele by mu mělo být umožněno se opět přihlásit a pokračovat v prohlížení webu tam, kde předtím skončil - a to bez ztráty dat.

Pravidla však nezapomínají ani na uživatele s různými typy záchvatů. Ty říkají, že žádný prvek na webu by neměl blikat více jak třikrát za sekundu, s výjimkou situace, kdy jsou tyto prvky dostatečně malé, mají nízký kontrast a neobsahují příliš mnoho červené barvy. (WebAim, 2015)

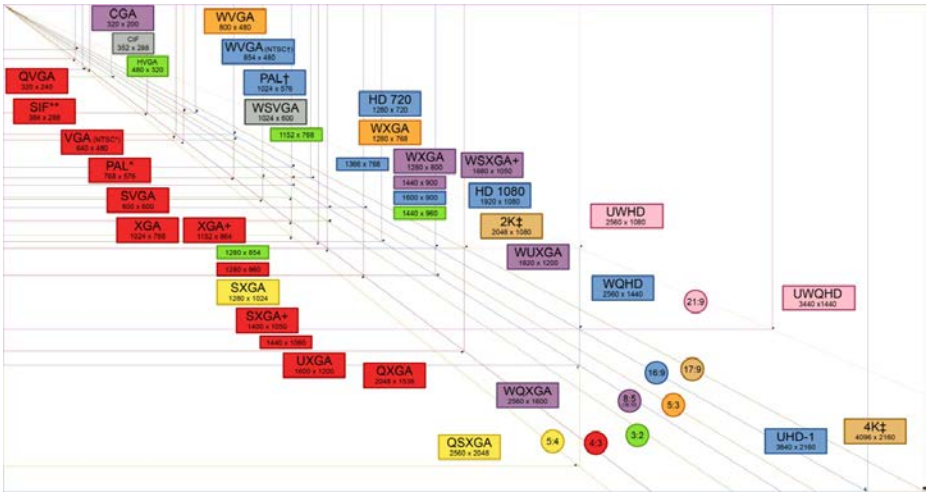
Součástí ovladatelnosti webu je i způsob navigace a pohybu po samotné stránce. V rámci těchto pravidel je uvažováno například používání tlačítek pro přeskakování bloků, které se na webu opakují. K tomu je možné využívat například přeskakování obsahu po nadpisech, které však musejí být správně definované. Dále je nutné definovat každé stránce vhodný a výstižný titulek, který usnadní ovladatelnost přepínání záložek v prohlížeči.

Veškerý obsah stránek by měl být dostupný minimálně dvěma různými způsoby. Těmi může být vedle klasické navigační lišty například průchod skrz mapu stránek, fulltextové vyhledávání, či vytvoření propojeného obsahu.

3.5 Zařízení pro přístup k webu a jejich rozlišení

Současné prostředí a přístup k prohlížení webu nemůže být rozmanitější. Díky značnému množství prohlížečů a ještě většímu množství zařízení, která poskytují připojení k internetu, se weboví vývojáři setkávají s problémem, jak zpřístupnit stránky všem uživatelům na všech platformách. V roce 2008 počet zařízení s přístupem na internet převýšil počet uživatelů. (Gasston, 2015)

Základním aspektem pro charakteristiku těchto zařízení je velikost jejich obrazovky a rozlišení, v němž mohou web zobrazovat. Do roku 2012 bylo nerozšířenějším rozlišením obrazovky mezi uživateli rozlišení 1024×768 pixelů. V následujícím roce se však na vrchol začalo dostávat notebookové rozlišení 1366×768 s poměrem stran 16:9. Dalším velmi oblíbeným rozlišením mezi uživateli je FullHD, tedy 1980×1080 pixelů. Tím se však vývoj nezastavuje. Dnes se může uživatel setkat i s daleko větším rozlišením. Ve skutečnosti však existuje 8 běžných poměrů stran ve 39 obvyklých rozlišeních. (Souček, 2015)



Obrázek 2 – Typy rozlišení obrazovek, zdroj: (Tichá, a další, 2014)

S rozlišením obrazovky jsou však spojeny další pojmy. Dnešní mobilní zařízení disponují obrazovkami s velmi vysokým rozlišením. Z toho důvodu je při tvorbě webových stránek třeba uvažovat o zobrazení nejen z pohledu velikosti jednotlivých prvků, ale také o tom, jak se na konkrétních zařízeních budou zobrazovat. Jeden a tentýž obrázek se na totožném rozlišení může jevit hezky na velikosti úhlopříčky 24“, nikoliv však na mobilním telefonu s 5“ obrazovkou.

Obrázek 3 - Obrázek 500×500 pixelů na FullHD rozlišení na 24" monitoru a 5" telefonu

3.5.1 Desktopové počítače a notebooky

Většina uživatelů se k internetu připojuje prostřednictvím svého počítače nebo notebooku. Díky tomu mají tyto zařízení stále největší podíl na trhu.

Z průzkumu společnosti IIBR uživatelé na stolních počítačích nejčastěji hledají informace a kontrolují internetové bankovnictví z pocitu většího zabezpečení. Počítače jsou využívány také ke zpracování většího množství textu, ať už jde o e-mailovou komunikaci, či psaní dokumentů. Jejich hlavní výhodou je kontrola nad obsahem, což je často vyžadováno při složitějších operacích, jako je například nákup v internetovém obchodě, správa redakčních systémů a podobně. (Gasston, 2015)

Desktopy a notebooky jsou mezi uživateli oblíbené především díky velikosti obrazovky a způsobu ovládání. Vývoj pracovního prostoru zaznamenal značný pokrok. Od dob, kdy se lidé k internetu připojovali na rozlišení 640×480 pixelů uplynulo již mnoho let. Dlouhou dobu bylo velmi oblíbené rozlišení 1024×768 pixelů na 15“ a 17“ CRT, později LCD, monitorech. To se poprvé objevilo již v roce 1990, ale mezi širokou veřejností se prosadilo až o pár let později. Nástupcem se stalo rozlišení 1280×1024, tedy SXGA. Dnešním standardem se pomalu, ale jistě stává Full HD rozlišení s 1920×1080 pixely. (Gasston, 2015)

S poklesem cen notebooků se do povědomí uživatelů dostal i poměr stran 16:9 a s ním dodnes nejrozšířenější rozlišení obrazovek 1366×768, které je pomalu vytlačováno Full HD displeji.

3.5.2 Mobilní telefony a tablety

Ještě do nedávna cílil web především na výkonné počítače se stabilním kabelovým či Wi-Fi připojením k internetu. Díky poklesu cen chytrých telefonů a tabletů jsou však počítače vytlačovány především z domácností. Právě tato obměna používaných zařízení byla hlavním impulsem pro rozvoj responsivního webdesignu.

Výběr těchto zařízení je opravdu veliký, od levných telefonů s jednoduchým webovým prohlížečem, přes střední třídu telefonů s optimalizovaným prohlížečem pro rychlé a datově úsporné prohlížení, až po drahé smartphony s plnohodnotným prohlížečem, který může směle konkurovat desktopovým aplikacím.

Velmi podobně jsou na tom i tablety. Díky jádru OS systému Windows 8.1 a 10 jsou tato zařízení vybavena plnohodnotnou aplikací pro prohlížení webu. U tabletů se systémem Android se uživatelé setkávají s odlehčenou verzí prohlížečů s optimalizovaným výkonem pro datové připojení a hardwarové vybavení.

Velký význam u přenosných zařízení hraje jádro WebKit. To dominuje na zařízeních s operačním systémem Android, iOS, nebo i Blackberry OS. Značný podíl na trhu mobilních prohlížečů má posléze společnost Opera s aplikací Opera Mini. (Market, 2016)

Vyvíjet webové stránky pro mobilní telefony a tablety s sebou nese značná úskalí především kvůli široké škále možných rozlišení obrazovek. Od levných 3,5“ smartphonů s rozlišením displeje 480×320 pixelů až po drahé 5,7“ telefony s retina displejem o rozlišení až 2560×1440. (Gasston, 2015) Podobně jsou na tom i tablety, u kterých je jediným rozdílem větší úhlopříčka obrazovky. Dále je nutné si uvědomit možnost otáčení těchto zařízení mezi variantou *portrait* a *landscape*.

Obrázek 4 - Možnost otáčení mobilních zařízení, verze *Portrait* a *Landscape*

Tvůrci webů by měli dále věnovat pozornost datové velikosti samotných stránek. Díky mobilnímu internetu se uživatelé připojují k webu prakticky odkudkoliv na světě. S tím jsou spojeny 2 hlavní problémy:

- Limity čerpání dat, neboli FUP
- Nedostatečný signál (pomalé připojení k internetu)

Z těchto důvodů je vhodné minimalizovat datovou náročnost samotných stránek, především ve variantě pro mobilní zařízení.

Není to však jen o technologiích a službách, které tyto přístroje uživatelům poskytují. Při vývoji webových stránek přístupných i z mobilních telefonů musejí vývojáři brát v úvahu i skutečnost, že jejich uživatelé používají svá zařízení v mnoha různých situacích, ať už je to za chůze, ve frontě v obchodě, v práci, ve výtahu, či v dopravním prostředku. Z toho důvodu nabírá pojem přehlednost a ovladatelnost stránek úplně nový rozměr. (Gasston, 2015)

3.5.3 Čtečky knih

S příchodem zařízení Amazon Kindle, určeného pro čtení elektronických knih, se začal rozmáhat i trend jednoduchého zobrazování webů, především s nabídkou elektronických knih pro tyto a jiné čtečky elektronických dokumentů. Díky elektronickému inkoustu jsou však tato zařízení omezena pouze na škálu odstínů šedé. Prohlížení webu na těchto zařízeních je tedy spíše nouzovým řešením.

3.6 Měrné jednotky na webu

S rozlišením a velikostí obrazovek je spojeno značné množství dalších pojmů od zadávání velikostí jednotlivých prvků až po četnost pixelů na palec. Rozměry elementů a fontů na webu se dají udávat ve značném množství různých jednotek, přičemž každý typ má jiné využití a způsob fungování. Při tvorbě webu se vývojáři setkávají s dvěma hlavními kategoriemi pro udávání velikostí. Těmi jsou velikosti písma a velikosti elementů.

3.6.1 Velikosti písma

Pro udávání velikosti písma se velmi často používají vedle klasických pixelů různé relativní a dopočítávané jednotky. Velikost jde tedy zadávat různě. Jednotlivé varianty hodnoty atributu *font-size* jsou znázorněny v následujícím seznamu:

- velikost písma prostřednictvím pixelů
  - `element { font-size: 50px; }`
- slovní zadání velikosti písma
  - `element { font-size: small; }`
  - `element { font-size: xx-large; }`
- procentuální zadání velikosti písma, které se odvíjí od velikosti rodičovského elementu, a alternativa v podobně jednotek *em*
  - `rodicovsky-element { font-size: 50px; }`

    - `element { font-size: 150%; }` – velikost bude tedy 75px
    - `element { font-size: 1.5em; }`
- relativní jednotky odvíjející se od nastavení velikosti písma kořenového elementu html
    - `html { font-size: 50px; }`
    - `element { font-size: 1.5rem; }` – velikost bude tedy 75px (W3Schools, 2015)

3.6.2 Velikosti elementů

Pro zadávání rozměrů se nejčastěji používají klasické měrné jednotky – pixely, případně relativní alternativa v procentech. Klasické zadávání rozměrů prvků může vypadat následovně (vlevo absolutní rozměr, vpravo relativní rozměr):

```
element {
  width: 500px;
  height: 250px;
}
```

```
element {
  width: 50%;
  height: 25%;
}
```

U definování rozměrů elementu lze vedle klasických absolutních jednotek v podobě pixelů a relativních jednotek používat i speciální nastavení velikosti odvíjející se od velikosti průhledu. Velmi užitečnou hodnotou jsou jednotky *vh* a *vw*, tedy *viewport height* a *viewport width*. Ty pracují s velikostí samotného průhledu zařízení. Samotné jednotky se udávají v procentuálním poměru vůči velikosti zobrazovaného okna.

```
element {
  width: 100vw;
  height: 100vh;
}
```

Takto nastavený element vyplní 100% průhledu jak na výšku, tak na šířku. (Snook, 2015)

3.6.3 Četnost pixelů a převodní poměry

Četnost pixelů na obrazovce se označuje zkratkou PPI, neboli *Pixel Per Inch* (Pixel na palec). Tento pojem je díky grafické aplikaci od společnosti Adobe s názvem Photoshop často také označován zkratkou DPI, tedy *Dot Per Inch*. Tato hodnota určuje, kolik pixelů je umístěno na délce jednoho palce. (Gasston, 2015)

Do nedávna byla standardní hodnota DPI rovna počtu 96 pixelů na jednom palci. Díky miniaturizaci však toto číslo stále roste. Telefon iPhone 3GS má DPI obrazovky 163 a jeho nástupce, tedy 4. generace, má DPI dokonce 326. U poslední verze iPhone 6s plus je DPI dokonce na hodnotě 401. (Apple, 2015)

Z toho důvodu je nutné rozlišovat pojmy *„fyzický pixel“* a *„virtuální pixel“*. Fyzické pixely určují přesné hardwarové rozlišení obrazovky. Tedy, pokud má obrazovka rozlišení 1920×1080, jedná se o fyzické rozlišení. Tyto rozměry jsou dále závislé na již zmíněné hustotě pixelů na palec. Vedle toho virtuální pixely jsou nezávislé na hustotě. Označují se DIP (z anglického *density-independent pixel*). V praxi se jedná o relativní jednotku rozlišení a jeden fyzický pixel se tedy může rovnat libovolnému množství těch virtuálních.

S tím je spojen další pojem, kterým je *poměr pixelů zařízení*, označovaný též DPR. V případě, že zařízení nemá tento poměr stanoven, je poměr roven 1:1, tedy jeden fyzický pixel se rovná přesně jednomu virtuálnímu. V době psaní této diplomové práce měla běžná zařízení s vysokým rozlišením obrazovky poměr 1 : 2. V praxi to vypadá následovně. (Gasston, 2015)

Obrázek 5 - DPR obrazovky, zleva 1:1, 1:1,5 a 1:2

Hodnota převodového poměru lze snadno zjistit prostřednictvím JavaScriptu a vlastnosti modelu DOM s názvem *devicePixelRatio* objektu *window*.

```
window.devicePixelRatio;
```

3.7 Responsivní a adaptivní web

Pojem *responsivní webdesign* se poprvé objevil v roce 2010 ve stejnojmenném článku pro web *A list Apart* Ethana Marcotteho. (Marcotte, 2010) Během pár let se tento

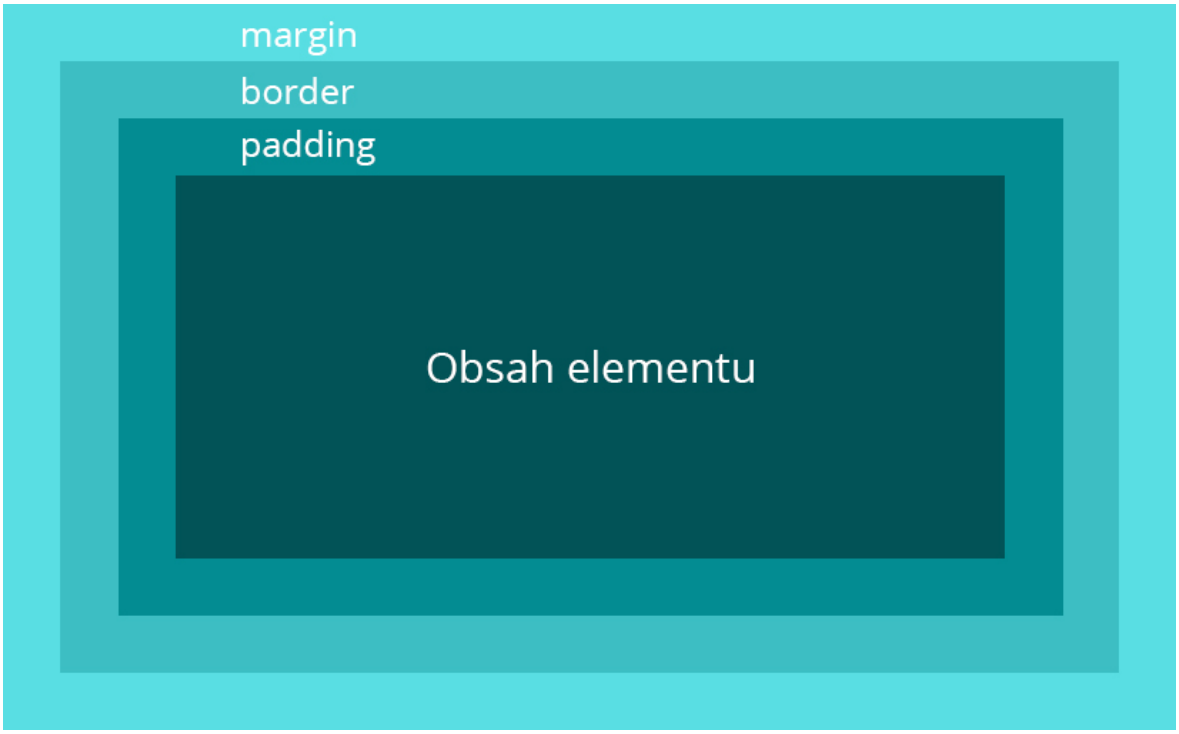
pojem stal neoddělitelnou součástí tvorby webových prezentací. Vedle responsivního designu se však objevil i další pojem, kterým je *adaptivní webdesign.* Cílem obou těchto metod je poskytnout návštěvníkovi stránek jednotný uživatelský prožitek napříč všemi dostupnými zařízeními. Naopak hlavním rozdílem mezi těmito dvěma pojmy je jejich pohled a přístup k tvorbě přizpůsobitelných webových stránek.

Obrázek 6 - Rozložení obsahu webu a jeho responzivní adaptace

Obě tyto koncepce vycházejí z definování zlomových bodů, které jsou realizovány prostřednictvím dotazů na medium, jež budou podrobně popsány dále v této práci. Zatímco adaptivní web na různých úrovních zlomových bodů využívá vždy jen pevné rozměry všech elementů, responsivní web pracuje s relativními jednotkami. Obě metody mají svá pro a proti a je na vývojáři, aby zvolil nejvhodnější kompromis mezi oběma přístupy.

Dle Petera Gasstona je efektivnější využívání právě relativních jednotek pro definování rozměrů elementů, tedy takzvaný *fluidní design*. Prvky na stránce se díky používání procentuálních jednotek přizpůsobují velikosti průhledu. Často zde však nastává problém například při oddělování jednotlivých prvků od sebe. Tyto tzv. oddělovače je však vhodné nastavovat na pevný rozměr. S tím přichází křížení responsivního a adaptivního přístupu k rozvrhování obsahu. (Gasston, 2015)

Box model kaskádových stylů nabízí značné možnosti, jak tohoto oddělení dosáhnout. Je však nutné si uvědomit princip fungování samotného box modelu, který je zobrazen na následujícím obrázku:



Obrázek 7 - Box model CSS

Z toho je patrné, že používání vlastností pro odsazení jednotlivých elementů samotné prvky zvětší.

```
element {
  width: 500px;
  margin: 10px;
  padding: 15px;
  border: 5px;
}
```

Element z výše zmíněného příkladu tak ve skutečnosti nebude mít šířku 500 pixelů, ale šířku ze vzorce:

```
width + margin-left + margin-right + padding-left + padding-right + border-left + border-right = 500 + 10 + 10 + 15 + 15 + 5 + 5 = 560px
```

A to samé pravidlo platí i při používání relativních jednotek.

```
element {
  width: 50%;
  margin: 10px;
  padding: 15px;
  border: 5px;
}
```

Šířka zde bude vycházet z 50% šířky rodičovského prvku + margin + padding + border. Je tedy poměrně složité definovat správné relativní rozměry tak, aby se při započtení box modelu CSS zobrazovaly přesně dle požadavků.

Tento problém řeší nová vlastnost CSS3 s názvem *box-sizing*. Pomocí ní lze definovat hranice, od kterých bude prohlížeč vypočítávat rozměry. Jednoduše řečeno lze touto vlastností měnit samotný box model. Může nabývat těchto hodnot:

- *content-box* – výchozí hodnota, která do rozměru prvku nezahrnuje žádný rozměr z vlastnosti margin, padding, či border
- *border-box* – do šířky elementu budou zahrnuty vnitřní odsazení (padding) a okraje samotného prvku (border), nikoliv však vnější okraje (margin)
- *initial* – nastaví element na výchozí hodnotu konkrétního elementu
- *inherit* – nastaví hodnotu na hodnotu rodičovského elementu (W3Schools, 1999-2016)

3.7.1 Mobile First

Jednou z nejefektivnějších metod tvorby responsivního designu je *Mobile First*. Ta říká, že prvotní styly se začnou definovat od verze pro nejmenší rozlišení a následně se postupně rozšiřují pro větší obrazovky a stolní počítače. S tím jsou spojeny zlomové body prostřednictvím dotazů na médium.

Díky tomu lze definovat různé šablony stylů, které pokryjí většinu dostupných zařízení. Luke Wroblevski, který je jedním z hlavních zastánců této metody, ve své knize Mobile First napsal (Wroblewski, 2011):

„Navrhování pro mobilní zařízení v první řadě neotvírá jen nové možnosti k růstu, ale může zlepšit celkový uživatelský prožitek z webových stránek nebo aplikací.“

Wroblevski ve stejné knize napsal také:

„Designéři, navrhujte nejprve pro mobily. Protože smartphony se prudce šíří mezi uživateli, protože nás díky svými omezeními nutí zaměřit pozornost na to nejdůležitější a protože rozšiřují naše možnosti.“

To rozvinul Martin Michálek na svém blogu VzhůruDolů. Celkově toto tvrzení shrnul do pár základních bodů. Uživatelé, prohlížející web na smartphonu:

- mají málo času
- nechtějí příliš klikat při cestě k požadovaným informacím
- nechtějí vyplňovat nesmyslně dlouhé formuláře
- nebudou trpěliví (Michále, 2015)

Hlavní myšlenkou při tvorbě responzivního designu není určování zlomových bodů podle rozlišení jednotlivých zařízení, ale soustředění se na čitelnost obsahu a volit zlomy podle toho, kdy se obsah webu začíná stávat nečitelným. S tím je spojen i počet textových znaků na řádek, který by se měl pohybovat v rozmezí 45 až 75 znaků, optimálním počtem je 66 znaků, což vychází z definice normostrany. (Gasston, 2015)

3.7.2 Nastavení webu pro responsivní zobrazení

Webové prohlížeče na mobilních zařízeních umožňují změnit velikost stránky tak, aby vyplnila celý prostor obrazovky. Toto se obvykle označuje jako průhled rozvržení. Výchozí nastavení průhledu u prohlížeče na operačním systému iOS je nastaveno na 980px, nejrozšířenější mobilní prohlížeč Opera Mini má průhled 850px a renderovací jádro systému Android dokonce pouhých 800px. (Craig Sharkie, 2015) To znamená, že element o velikosti 200×200px vyplní na různých zařízeních odlišnou plochu obrazovky a ve výchozím nastavení se nezobrazí v poměru 1:1 k originálním rozměrům.

Tento problém se rozhodla vyřešit společnost Apple při vývoji svého prohlížeče Safari. Implementací tagu *viewport* zajistila, že prohlížeč stránku nepřizpůsobuje automaticky, ale má předem nastaveno, v jakém měřítku bude obsah načten.

Zápis značky pro definování viewportu vypadá následovně:

```
<meta name="viewport" content="direktiva, direktiva, …">
```

Mezi parametry této značky lze zanést několik alternativ. První z nich je direktiva *width*. Ta umožňuje nastavit zobrazení stránky na konkrétní šířku, či šířku zařízení.

```
<meta name="viewport" content="device-width">
```

Toto nastavení znamená, že se velikost stránky přizpůsobí rozměrům obrazovky.

```
<meta name="viewport" content="width=200px">
```

Při použití přesného nastavení rozměru se šířka stránky přepočítá tak, aby vyplnila obrazovku v daném poměru. V případě, že bude například nastavena hodnota 200px, ale šířka zařízení bude 540px, veškeré elementy na stránce se změní v poměru 2,7.

Obrázek 8 - Viewport s parametrem width=200px, 500px a s parametrem device-width

Další možností nastavení je direktiva *height*, tedy výška zobrazení. V praxi funguje totožně jako její protějšek a není tedy nutné ji více popisovat.

Součástí meta značky *viewport* je i možnost zakázání přibližování či oddalování stránky. Zápis vypadá následovně:

```
<meta name="viewport" content="user-scalable=no">
```

Tento kousek zdrojového kódu zabrání uživateli zvětšovat či zmenšovat zobrazení. Je však důležité si předem promyslet, zdali je odebrání této možnosti vhodné, protože značné množství uživatelů by to nemuselo snést.

Dále je možné nastavit počáteční úroveň změny velikosti zobrazení. Rozmezí je od 10% do 1000% (zápis desetinným číslem, tedy 0,1 až 10,0).

```
<meta name="viewport" content="initial-scale=0.8">
```

Znamená to tedy, že v momentě nastavení hodnoty 0,8 se web o šířce 1000 pixelů zmenší na 80% skutečné velikosti, tedy na 800px. S tím jsou spojeny další dvě vlastnosti, které uživateli umožňují maximální (*maximum-scale*) a minimální (*minimum-scale*) zobrazení stránky. (Kadlec, 2014)

Nevýhodou meta tagu je nutnost jeho použití skrz všechna zařízení. Tedy v momentě, kdy autor stránek nastaví 1,5 násobné zvětšení, web bude takto nastaven napříč všemi zařízeními od malých telefonů až po velké monitory stolních počítačů. (Gasston, 2015)

Alternativou, která tento problém řeší, je nastavení průhledu prostřednictvím kaskádových stylů. Toto pravidlo se nazývá `@viewport` a jeho hlavní výhodou je možnost propojení s mediálními dotazy, které budou dále v této práci popsány. Tato metoda umožňuje nastavit různé vlastnosti průhledu na základě předem definovaných podmínek, jako je například šířka zobrazení. Podpora tohoto pravidla však stále není dostatečně rozsáhlá. (Craig Sharkie, 2015)

3.8 Možnosti zadávání dat a ovládání

3.8.1 Klávesnice, myš a touchpad

Nejčastěji používaná ovládací zařízení webu jsou bezesporu myš (případně touchpad) a klasická hardwarová klávesnice s QWERTY rozložením kláves. I to je jeden z důvodů, proč jsou klasické počítače stále nejpoužívanější zařízení, sloužící k přístupu na internet. (Craig Sharkie, 2015)

Hardwarová klávesnice umožňuje snadné a rychlé zadávání textu, přepínání záložek prostřednictvím klávesových zkratek, přechody po stránce pomocí šípek, kláves PageDown a PageUp, či skoky ze začátku na konec pomocí kláves Home a End.

Vedle toho největší výhodou při používání myši či touchpadu, respektive klasického kurzoru, je možnost využití *hover* efektu. Tedy situace, kdy uživatel kurzorem najede nad nějaký prvek a ten se na základě předem definovaných kaskádových stylů, či *OnMouse* efektu napsaném v jazyce JavaScript nějak změní. Tato funkce je velmi nápomocná pro zefektivnění přehlednosti webových stránek, či pro zviditelnění některých prvků.

Obrázek 9 - Hover efekt při použití myši

3.8.2 Dotyková obrazovka

Alternativou ke kurzoru myši a efektu po najetí jsou takzvané *Touch events,* tedy události, které proběhnou po nějakém kontaktu prstu s obrazovkou. Ve specifikaci Touch Events Level 1 jsou čtyři základní události:

- *Touchstart*, tedy událost spuštěná při doteku prstu na konkrétním elementu
- *Touchend* je přesný opak události *Touchstart*
- *Touchcancel* nastává při přerušení kontaktu s obrazovkou nebo opuštění definované oblasti
- *Touchmove* je událost, která je definována pohybem prstu mezi 2 body.

S rozšířením Touch Events Level 2 jsou do specifikace zaneseny dále:

- *Touchcenter* nastává tehdy, když se prst na obrazovce přesune nad konkrétní objekt
- *Touchleave* je opakem události *Touchcenter*, nastává tedy v momentě, kdy se prst pohybující se po obrazovce dostane mimo vytyčenou oblast. (Schepers, 2013)

Veškeré tyto události jsou zanášeny do Touch Listu. Vzhledem k tomu, že dotykové obrazovky velmi často disponují funkcí multitouch, tedy dotyk více prsty zároveň, jsou do tohoto seznamu zanášeny hodnoty v podobě:

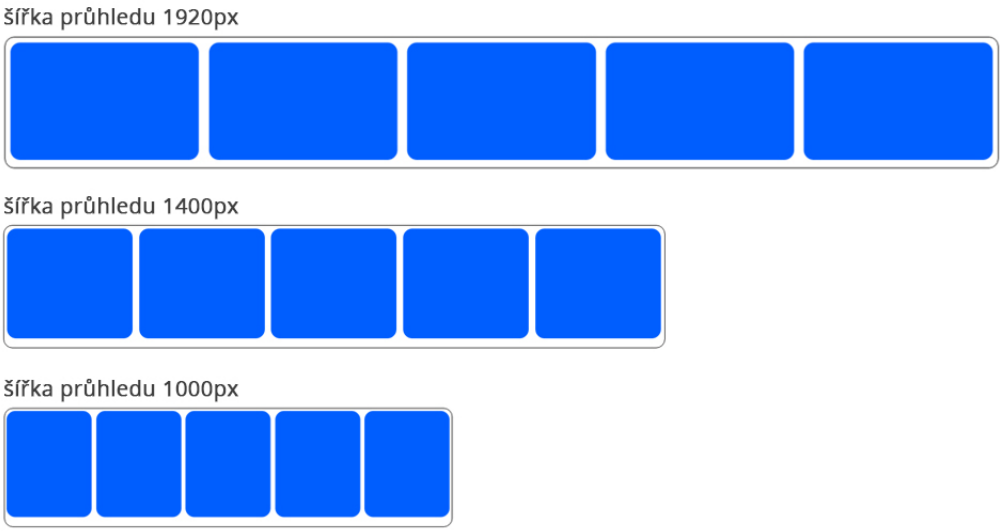
- 1 prst – informace o dotyku 0
- 2 prsty – informace o dotycích 0 a 1
- 3 prsty – informace o dotycích 0, 1, 2
- atd.

3.9 Vývoj kaskádových stylů z pohledu responsivního designu

3.9.1 Fluidní mřížky

Jedním z přístupů k responsivnímu designu je využití tzv. fluidních mřížek. Jejich původcem jsou tiskové mřížky, které mají pevnou šířku a určují rozvržení textu. Základní myšlenkou tohoto principu je rozvržení obsahu do několika sloupců a řádků, které se přizpůsobují svou šířkou a umístěním konkrétní velikosti obalujícího elementu, ať už je to samotná zobrazovací plocha, nebo nějaký jiný přesně nebo relativně definovaný box.

Výhodou fluidních mřížek je, že stránka automaticky vyplňuje celou požadovanou oblast obsahem. Základem je využívání relativních jednotek pro definování rozměrů jednotlivých elementů.

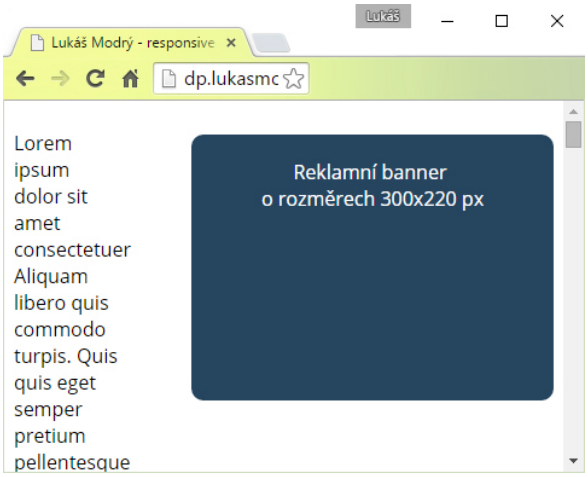


Obrázek 10 - Fluidní mřížka

Problém však nastává v momentě, kdy jsou fluidní mřížky kombinovány s fixně definovanými elementy, které se používají například jako levý panel pro reklamní bannery apod. Přestože takové použití může na většině dostupných zobrazení vypadat dobře, existují situace, kdy je tomu naopak. (Sucharda, 2013)

Jednou takovou situací je roztažení prohlížeče do enormně velkých rozměrů. Je třeba si uvědomit, že vhodný počet znaků na řádek, aby byl text pohodlně čitelný a přehledný, je 66. Při zvětšení elementu za určitou hranici však vzniká více prostoru a obsah stránky se tak stává špatně čitelným. (Dobrý web, 2009)

Dalším krizovým momentem je přesný opak, tedy zúžení prohlížeče do příliš malých rozměrů, kde se na řádek vejde jen pár slov, což je nejen nevzhledné, ale i nepohodlné ke čtení. V takové chvíli často reklamní banner přebíjí hlavní obsah stránky, nehledě na to, že se uživateli často zobrazí horizontální scrollbar. V případě, že pro pevně definovaný element nezbude dostatek prostoru, odskočí dolů, což je často nežádoucí.



Obrázek 11 - Nevhodné využití fluidních mřížek v kombinaci s pevným panelem

3.9.2 Media queries

Pro doplnění fluidních mřížek vznikl nástroj s názvem *mediální dotazy*, neboli *media queries*. Ty fungují na principu jednoduché podmínky s výstupem *true / false*. Pokud je podmínka splněna, načtou se konkrétní předdefinované šablony stylů napsané pro tuto situaci. (Kadlec, 2014)

```
@media [not|only] typ [and] (výraz)
{ Blok předdefinovaných kaskádových stylů }
```

Z obecného zápisu dotazu je patrné, že se skládá z více částí:

- *Typ media* určující zařízení, pro které je dotaz napsán
- *Mediální výraz* je parametr pro definování konkrétní podmínky, která je buď splněna, či nikoliv
- *Výroková logika* pro komplikovanější dotazy složené z více podmínek
- *Blok předdefinovaných kaskádových stylů*, tedy styly, které jsou aplikovány, pokud je splněna podmínka definovaná v dotazu

Web je velmi specifické prostředí, které se neustále vyvíjí a rozšiřuje. Jeho největší výhodou a zároveň prokletím je možnost obsluhovat ho na velmi různorodých zařízeních. Ta nejsou omezena pouze na obrazovky počítačů, ale i na mobilní telefony, tablety, televize, nebo dokonce speciální přístroje pro zobrazování textu pomocí Braillova písma. Právě mediální dotazy slouží k jejich roztřízení a úpravě vzhledu pro jednotlivé druhy zobrazení. Nalezení nejvhodnější kombinace pro sestavení dobrého mediálního dotazu však není úplně jednoduché. Zoe Mickley Gillenwater o tomto problému napsal příspěvek

s názvem *„Essential Considerations for Crafting Quality Media Queries“* (Gillenwater, 2011), v němž uvádí:

„Navrhování rozvržení webových stránek s pomocí dotazů na médium je složitý proces. Musíte s nimi počítat už od začátku a rozhodovat se, jak je nejlépe začlenit tak, aby dávaly smysl. Málokdy existují odpovědi na to, co je zaručeně správné nebo špatné.“

Lze je používat buď přímo v hlavičce webové stránky při načítání souborů s kaskádovými styly, nebo přímo v předpisu kaskádových stylů. Je otázkou každého jedinečného projektu, kterou alternativu zvolit. (Kadlec, 2014)

Samotné zápisy takových mediálních dotazů vypadají následovně. V případě externího zápisu přímo v hlavičce webu jsou atributem elementu link:

```
<link href="style.css" media=" … ">
```

Druhým způsobem je zapsání dotazu přímo v souboru kaskádových stylů:

```
@media typ logický-výraz (další podmínky) { … }
```

V obou případech jsou stahovány veškeré styly pro případ, že by se změnil způsob zobrazení. Taková situace může nastat například překlopením mobilního telefonu, kde se mění rozlišení z režimu *portrait* na režim *landscape*, další takovou situací může být například změna velikosti okna prohlížeče. V případě, že by konkrétní styly nebyly okamžitě k dispozici, webová stránka by se musela obnovovat a znovu vyhodnocovat mediální dotaz. Takový web by nepůsobil důvěryhodně a použitelně. (Craig Sharkie, 2015)

Zásadním rozdílem je počet HTTP požadavků na server. Zatímco u dotazu, který je použit přímo v předpisu kaskádových stylů, je požadavek pouze 1 a všechny styly jsou načteny najednou, v hlavičce webové stránky je počet dotazů roven počtu jednotlivých souborů s předpisem stylů. To způsobuje problémy při načítání stránek na mobilních sítích, které mají často horší odezvu serveru a stahování souborů je tak náročnější. Proti tomu však centralizování stylů a dotazů do 1 souboru může vytvořit podstatně datově náročnější dokument. (Kadlec, 2014)

Mediální typy

Mediální typy slouží k roztřízení zařízení podle jejich funkce. Dotaz na základě vyhodnocení podmínky o zařízení, na kterém je web zobrazován, předá prohlížeči informaci, zda má načíst konkrétní kaskádové styly, nebo je vynechat. Takto zapsané

pravidlo bude nadefinované styly aplikovat jen na konkrétních zařízeních zvoleného (Gasston, 2015)

Dnešní dotazy znají 10 různých typů zařízení, pro které mohou zobrazovat styly:

- *all* neboli typ pro shrnutí všech zařízení, tedy instrukce, aby se styly načetly na všech zařízeních
- *braille* pro zařízení, která dovedou grafický výstup převést do Braillova písma
- *embossed* pro tisk v Braillově písmu
- *handheld* pro malá zařízení, která uživatel drží v ruce. Tedy mobilní telefony, tablety apod.
- *print* pro dokumenty určené k tisku
- *projection* pro zobrazování webových stránek prostřednictvím projektoru
- *screen* je typ pro zobrazení na klasických obrazovkách počítačů a notebooků
- *speech* pro zařízení, převádějící text na zvukový výstup
- *tty* pro zařízení, která využívají neproporcionální znaky
- *tv* pro televize (Schools, 2015)

Na příkladu níže je ukázán zápis 2 mediálních dotazů, kde v případě, že uživatel webovou prezentaci zobrazuje na klasickém monitoru, budou veškeré odstavce napsány červeným písmem. Zatímco pokud se pokusí tento web vytisknout, všechny odstavce budou černé, což může být využito například pro ušetření barvy v tiskárně. Třetí zápis na obou zařízeních nastaví velikost písma na 20px.

```
@media screen {
  p { color: red; }
}
@media print {
  p { color: black; }
}
@media screen, print {
  p { font-size: 20px; }
}
```

Totožná alternativa zápisu v hlavičce webu:

```
<link href="style-1.css" rel="stylesheet" media="screen">
<link href="style-2.css" rel="stylesheet" media="print">
<link href="style-3.css" rel="stylesheet" media="screen, print">
```

To je však pouhá jednoduchá ukázka a kompletní zápis stylů může být podstatně složitější. Některé elementy mohou být přebarveny, jiné skryty nebo zobrazeny úplně jinak. Možností je nepřeberně.

V reálné praxi jsou však nejpoužívanějšími typy *all*, *screen* a *print*. To je zapříčiněno tím, že většina vývojářů ve všech mediálních dotazech využívala právě typ *screen* a tomu se pak museli přizpůsobit distributoři webových prohlížečů.

Mediální typ není povinnou součástí syntaxe. Pokud není definován, měl by být automaticky nastaven na *all*. Praxe však ukázala, že ne vždy to funguje. (Gasston, 2015)

Mediální výrazy

Vedle mediálních dotazů jsou součástí syntaxe také mediální výrazy, které testují další parametry zobrazení a vyhodnocují, zdali nastal daný sled událostí, či nikoliv. Existuje mnoho typů výrazů, avšak při vývoji webových prezentací si tvůrci vystačí pouze s některými z nich.

- *width* pro vyhodnocení šířky viewportu zobrazení
- *height* jakožto opak k šířce k popisu výšky zobrazení
- *device-width* pro popis šířky renderovacího prostoru zařízení
- *device-height* obdobně pro popis výšky renderovacího prostoru
- *orientation*, který popisuje orientaci zařízení, tedy jestli je obrazovka na výšku nebo na šířku, což je nejčastěji využíváno u chytrých telefonů a tabletů
- *aspect-ratio* pro popis poměru stran viewportu
- *device-aspect-ratio* obdobně pro popis renderovací oblasti zařízení
- *color* popisující, zdali je zařízení barevné či nikoliv, například u čteček elektronických knih
- *color-index* pro počet položek ve vyhledávací tabulce barev zařízení, vrací hodnotu počtu barev
- *monochrome* pro počet bitů na pixel u monochromního zařízení, vrací hodnotu počtu bitů (například 8)
- *resolution* pro vrácení hodnoty počtu obrazových bodů na palec, či centimetr
- *scan* pro skenovací proces televizních a jiných podobných zařízení, vrací hodnotu progressive / interlace
- *grid* vrací, zdali je zařízení mřížkové (1) nebo bitmapové (0)

Nevýhodou značné části těchto výrazů je jejich podpora u jednotlivých prohlížečů. (Salvet, 2009)

Výroková logika v mediálních dotazech

Pro spojení jednotlivých mediálních typů a výrazů jsou dotazy obohaceny o základní výrokovou logiku. Využívání logických výrazů není povinné, přidává to však dotazům značnou míru variability.

Pro testování současné platnosti více kritérií existuje výraz `AND`. V příkladu je znázorněno spojení klasické obrazovky a minimální šířky zobrazení 1000 pixelů. Pro zobrazení bloku stylů musí být splněna veškerá kritéria zapsaná v dotazu.

```
@media screen and (min-width: 1000px) { … }
```

Oproti tomu může často být požadavkem splnění alespoň jednoho kritéria z mnoha. K tomu slouží logický výraz `OR`, který se však zapisuje čárkou.

```
@media screen and (min-width: 1000px), screen and (max-width
1500px) { … }
```

Takový zápis definuje způsob zobrazení pouze pro rozlišení v intervalu 1000 až 1500 px. Součástí výrokové logiky mediálních dotazů je negace, tedy `NOT`. To však neguje celý mediální dotaz, nikoliv pouze jeho části. V příkladu se styly nezobrazí, pokud uživatel stránku prohlíží na klasické obrazovce a velikosti průhledu s minimální šířkou 1000 pixelů.

```
@media not screen and (min-width: 1000px) { … }
```

Některé starší webové prohlížeče však mediální dotazy nepodporují. Umí reagovat pouze na mediální typ. To zapříčiní, že se tyto prohlížeče snaží stahovat i styly, které uživatel nemá vidět. Pro ošetření této situace slouží výrok `ONLY`.

```
@media only screen { … }
```

Pokud zařízení mediální dotaz nepodporuje, bude tento řádek zcela ignorovat. V opačném případě na něj zareaguje stejně, jako by zde bylo napsáno:

```
@media screen { … }
```

Mediální dotazy prostřednictvím JavaScriptu

Mediální dotazy jsou bezesporu velmi praktické. Mají tak i svou alternativu v jazyce JavaScript. Lze tedy volat funkce a načítat další skripty pro zlepšení zobrazení webové prezentace na různých zařízeních.

Hlavní metodou pro mediální dotazy je `matchMedia()`. Ta umožňuje volat dotazy ve formě textového řetězce, viz následující příklad:

```
matchMedia('sreen and (min-width: 800px)');
```

Tato metoda vypíše do konzole tzv. *MediaQueryList*, jehož součástí je vlastnost *matches* s hodnotou *true/false*, kterou lze využít pro sestavení jednoduché podmínky:

```
Var mq=window.matchMedia('sreen and (min-width: 800px)');
if(i) { foo(); } else { //udělat něco jiného }
```

Takto jdou pomocí jazyka JS volat různé styly a soubory stylů do webové stránky. (Gasston, 2015)

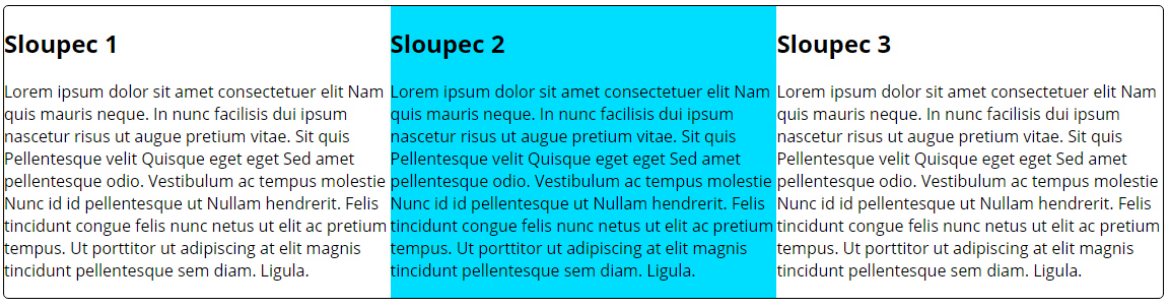
3.9.3 Flexbox

Specifikace CSS3 přináší novou hodnotu pro vlastnost *display*. Ta je vhodná především při tvorbě obsahově složitějších struktur jako jsou různé registrační formuláře, webové aplikace, či obsáhlé články.

Jádrem tohoto modulu je firefoxový jazyk pro rozvrhování obsahu XUL. Podstatou tohoto nastavení elementu je možnost flexibilní změny velikostí a uspořádání elementů. Ty díky této vlastnosti podstatně lépe vyplňují vytyčený prostor. Zápis vypadá následovně:

```
element { display: flex; }
```

V praxi funguje vlastnost tak, že se aplikuje na obalový element a všichni potomci se následně automaticky přizpůsobí svému nadřazenému elementu bez nutnosti přidávání vlastností *float*, *margin*, apod. jak tomu bylo doposud zvykem. (Gasston, 2015)



Obrázek 12 - Použití flexboxu v praxi

Dceřiné elementy jsou automaticky srovnány do řádku vedle sebe bez použití floatování pro určení pozic jednotlivých sloupců. Rovnání elementů lze však změnit další vlastností *flex-direction* na obalový element. Ta může nabývat těchto hodnot:

- *flex-direction: row;* je výchozí hodnota nastavená u této vlastnosti a zajišťuje uspořádání prvků do řádky
- *flex-direction: column;* srovnává elementy do sloupce, funguje tedy obdobně jako *display: block;*
- dále podobné hodnoty *row-reverse* a *column-reverse* pro snadné obrácení orientace elementů
- hodnota *initial* pro vrácení hodnoty do výchozího stavu
- hodnota *inherit* jakožto dědění z rodičovského element

Další možností pro srovnání elementů je vlastnost *order*, která se aplikuje na dceřiné elementy, nikoliv na obal. Ta funguje na principu sortování prvků prostřednictvím přidělení řadového čísla. Položky jsou řazeny od nejmenší hodnoty po největší a seskupeny. Na příkladu při použití struktury z obrázku výše:

```
<div class="rodic">
  <div class="potomek1"> … </div>
  <div class="potomek2"> … </div>
  <div class="potomek3"> … </div>
</div>
.potomek1 { order: 1; }
.potomek2 { order: 3; }
.potomek3 { order: 2; }
```

Po aplikování těchto stylů bude pořadí dceřiných elementů *potomek1*, *potomek3* a nejvíce vpravo bude *potomek2*.

Srovnané elementy je však potřeba nějak umístit, aby byl jejich obsah přehledný. K tomu slouží další vlastnost *justify-content*. V momentě, kdy jsou dceřiné elementy obsahově menší než jejich obal, prohlížeč vezme zbývající prostor a rozdělí ho podle definice mezi tyto elementy. Způsob rozdělení je závislý i na nastavení orientace prvků *flex-direction*, následující popis se bude vztahovat ke klasickému českému rovnání textu zleva-doprava. Styly 3. generace znají tyto možnosti: (Coyier, 2015)

- *flex-start* je výchozí hodnotou, která veškerý zbývající prostor nechá vpravo od obsahu
- *center* rozděluje prostor rovnoměrně na obě strany od krajních elementů
- *space-between* rozdělí prostor rovnoměrně mezi jednotlivé elementy, na jejich začátek a konec však nedá nic

- *space-around* rozdělí prostor rovnoměrně zleva i zprava každého dceřiného elementu

Pro elementy s nastavením *flex-direction: column;* jsou ve specifikaci další atributy:

- *flex-end, center, baseline* které fungují obdobně, jen ve vertikální orientaci

Tím je vyřešeno uspořádání elementů. Je však třeba jim přidat flexibilitu, tedy schopnost přizpůsobovat se. K tomu slouží série vlastností *flex-grow, flex-shrink* a *flex-basis*. Jejich hodnoty lze také zapsat do jednoho řádku pod vlastnost *flex*. Používají se u dceřiných elementů a vypadají následovně:

```
element { flex: 1 2 100px; }
```

nebo

```
element {
  flex-grow: 1;
  flex-shrink: 2;
  flex-basis: 100px;
}
```

Vlastnost *flex-basis* funguje podobně jako klasický *width*, je jí však nadřazená. Její funkcí je nastavení upřednostňované šířky elementů a zbývající prostor přidělí podle předdefinovaného poměru, který zajišťují ostatní zmíněné vlastnosti. Ty se navzájem doplňují.

Vlastnost *flex-grow* rozděluje zbývající prostor v obalovém elementu v definovaném poměru k dceřiným prvkům. Tedy pokud všechny budou mít nastavenou hodnotu této vlastnosti na 1, bude tento poměr 1:1:1:atd. (Gasston, 2015)

Tento poměr však může být libovolný pro jednotlivé elementy, což je znázorněno na následujícím příkladu:

```
.rodic { width: 600px; }
.potomek1 { flex: 1 2 150px; }
.potomek2 { flex: 1 2 150px; }
.potomek2 { flex: 3 4 150px; }
```

V tomto příkladu je nastaven poměr 1:1:3 a přebytečný prostor obalového elementu je 150px. Znamená to tedy, že tento prostor bude v poměru 30px:30px:90px rozdělen mezi potomky, kteří tak vyplní 100% svého rodiče. (Stojanov, 2015)

Přesně obráceně funguje vlastnost *flex-shrink*. V případě, že potomci překročí rozměr, kterým disponuje jejich rodič, tato funkce elementy naopak zmenší v daném poměru. Ten je ve výše zmíněném případě nastaven na poměr 2:2:4, tedy na 1:1:2. V momentě, kdy bude parametr *flex-basis* nastaven na 250px, bude v takovém případě přebytečných 150px odebráno v poměru 37,5px:37,5px:75px.

Občas nastane situace, kdy se i s výše zmíněnými vlastnostmi obsah nevejde do svého obalu a je třeba ho zalomit na další řádek. K tomu slouží poslední vlastnost k nastavení boxu, *flex-wrap*. Ta může nabývat hodnot:

- *nowrap* – boxy bez zalomení, výchozí hodnota
- *wrap* – zalomení na nový řádek
- *wrap-reverse* – zalomení na nový řádek v obráceném pořadí
- *initial* - pro vrácení hodnoty do výchozího stavu
- *inherit* – dědění z rodičovského elementu (Gasston, 2015)

3.9.4 Grid layout

Základem tohoto nástroje jsou mřížky, které využívali tiskaři již ve středověku. Stávající weboví vývojáři se snaží princip mřížky simulovat vlastností *float*, tedy prostřednictvím obtékání objektů, které doplňují klasickým vnitřním a vnějším odsazením od okrajů. Tento postup je však u složitějších struktur velmi náročný a nepřesný a často vzniká množství chyb způsobených nepřesným vykreslením.

Návrh mřížek, zanesený do 3. generace kaskádových stylů, přináší řešení v podobě nové možnosti pro vlastnost *display*. Tou je hodnota *grid*. Ta je v době psaní této práce součástí návrhu a není tak podporována téměř v žádném stávajícím webovém prohlížeči. K listopadu 2015 je podle webu *CanIUse.com* podpora tohoto nastavení jen v prohlížeči Internet Explorer 10+ a Microsoft Edge. (Deveria, 2016)

Základem mřížkového layoutu je definování obalového elementu, kterému je nastaven počet řádků a sloupců a jejich rozměry, či proporce. Samotný obalový element určuje hranice celé mřížky. Pro ukázku zde budou vypsány standardní zápisy vlastnosti bez použití různých vykreslovacích jader jednotlivých prohlížečů. (Gasston, 2015) Takový zápis vypadá například následovně:

```
obalovy-element {
  display: grid;
  grid-columns: 40% 40% 20%;
  grid-rows: 10% auto 30%;
}
```

Vývojáři však nejsou omezeni pouze na klasické délkové jednotky, na které jsou zvyklí. Specifikace přináší novou hodnotu *fr* pro definici rozměr. Ta rozděluje obsah na určité frakce, které fungují na principu rozdělení prostoru v definovaném poměru. (Gaebel, 2014) Předchozí zápis poměru 1 : 2 : 1 by vypadal následovně:

```
obalovy-element {
  display: grid;
  grid-columns: 2fr 2fr 1fr;
  grid-rows: 10% auto 30%;
}
```

Význam používání frakční jednotky získává význam až v momentě propojení s fixními jednotkami, kde je například potřeba zachovat přesný rozměr sloupce pro reklamní banner a zbývající prostor rozdělit v určitém poměru.

Rozvrhování obsahu do mřížky je nutné si předem důkladně rozmyslet a rozhodnout se kolika sloupcový / řádkový grid bude potřeba. Standardem se staly 12, či 16 sloupcové layouty pro rozvržení obsahu. Takový zápis by však byl velmi zdlouhavý. Proto je v momentě opakování stejného nastavení ve specifikaci připravena možnost *repeat()*, která umožňuje opakovat určitý kus kódu v definovaném počtu. (Gasston, 2015)

```
obalovy-element { grid-column 1fr repeat(16, 10px, 1fr);
```

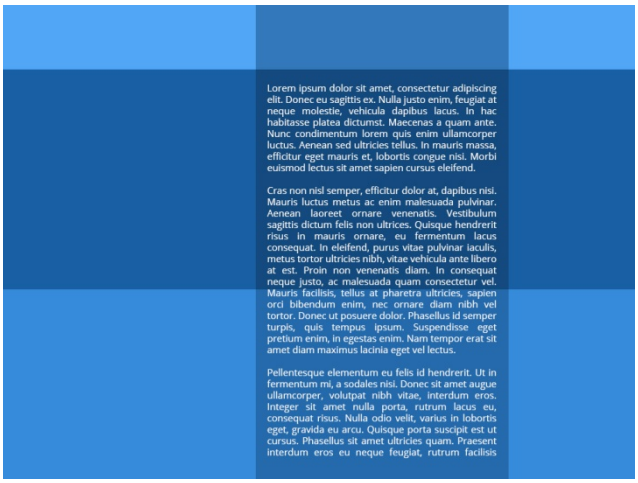
Samotný obsah je pak do tabulky umisťován prostřednictvím souřadnic mřížky. Pro umístění obsahového elementu je tedy třeba nastavit mu jeho horizontální a vertikální polohu. S tím je spojena i možnost umístit obsah přes více sloupců či řádků zároveň, podobně, jako je tomu u vlastností *colspan* a *rowspan* u klasických HTML tabulek. (Andrew, 2015)

```
element {
  grid-column: číslo-slopce;
  grid-row: číslo-řádku;
  grid-column-span: počet sloupců pro sloučení;
  grid-row-span: počet řádků pro sloučení;
}
```

V případě, že bude potřeba umístit například odstavec textu do prostřední buňky mřížky 3x3 a požadavkem bude, aby obsah přetekl o řádek níže, zápis bude vypadat následovně:

```
p {
  grid-column: 2;
  grid-row: 2;
  grid-rows-span: 2;
}
```

Ve spojení s předchozím příkladem pro vytvoření mřížky o rozměru 3x3 by tabulka a následné umístění zvoleného odstavce vypadaly přibližně následovně:



Obrázek 13 - Grid layout a umístění prvku do mřížky

Obsah lze dále různě zarovnávat na úrovni samotného řádku či sloupce mřížky. K tomu slouží 4 různé hodnoty atributů *grid-row-align* a *grid-column-align*:

- *end* – zarovnání od konce
- *start* – zarovnání od začátku
- *center* – zarovnání na střed
- *stretch* – roztažení položky na celou výšku / šířku

V praxi je používání mřížkového layoutu velmi podobné zastaralé metodě rozvrhování obsahu pomocí tabulky. Při složitějších strukturách může být velmi složité správné rozvržení tabulky a především prvků na správná místa. (Gasston, 2015)

3.10 Práce s obrázky

3.10.1 Element image

Pro responsivní práci s obrázky je v HTML a CSS několik různých alternativ. Jednou ze základních metod je element *img*. HTML5 tento prvek obohacuje o nové atributy, které podporují právě responsivní zobrazení obrázků. Těmi jsou atributy *srcset* a *sizes*.

Atribut *srcset* udává sadu obrázků, které se načtou při různých rozlišeních okna. K tomu využívá dvou desktriptorů, *w* a *x*.

```
<img src="maly.png" srcset="s.png 600w, m.png 1024w, l.png 1920w">
```

Takto nastavený element tedy načte obrázek *s.png* při velikosti okna 0 - 600px, ve velikosti 600 až 1024px zobrazí obrázek *m.png* a pro větší rozlišení načte největší variantu *l.png*. Pokud prohlížeč nové atributy nepodporuje, zobrazí se obrázek nastavený ve starém atributu *src*. Druhý deskriptor, *x*, určuje, jaký obrázek se má načíst při rozdílném *devices-pixel-ratio*.

```
<img srcset="s.png, l.png 2x" alt="popisek obrázku">
```

Takto zapsaný element zobrazí obrázek *s.png* při normálním poměru pixelů, a obrázek *l.png* při pixel ratiu 2 a větším.

Druhý atribut, *sizes*, určuje, jak velký bude obrázek při různé velikosti okna.

```
<img … sizes="(min-width: 768px) 300px, 100vw">
```

Tento zápis zobrazí obrázek široký 300px při rozlišení větším jak 768px. Pokud však velikost průhledu bude menší, tak obrázek vyplní celou šířku okna. (Michálek, 2015)

3.10.2 Element picture

Novým tagem HTML5 pro zobrazování obrázků na webu je `<picture>`. Funkce této značky je založena na mediálních dotazech.

Samotný prvek se skládá z 3 částí:

- `picture` – obalový element pro zobrazení obrázku
- `source` – seznam různých sad obrázků
- `img` – záložní zdroj obrázku při nepodporování prohlížečem (Gasston, 2015)

```
<picture alt="popisek obrázku">
  <source media="(min-width: 600px)" srcset="m.png">
```

```
    <source media="(min-width: 1024px)" srcset="l.png">
    <img src="s.png" alt="popisek obrázku">
</picture>
```

Na výše znázorněném příkladu jsou vytvořeny 2 různé mediální dotazy. První načte obrázek *m.png* při rozlišení okna v intervalu 600 až 1021 pixelů. Druhý načte obrázek *l.png* při rozlišení větším jak 1024px. Při nepodpoře elementu prohlížečem, případně při nesplnění žádného mediálního dotazu (v tomto případě průhled s menším rozlišením, jak 600px) se načte obrázek nadefinovaný ve značce *img*.

Element *source* může být dále obohacen o atribut *srcset*, který zde funguje obdobně jako u značky *image*. Lze tedy například zvolit různé obrázky při stejném rozlišení, ale jiném DPI. Element *picture* je vhodný pro zobrazování různých obrázků (například jiného výřezu) nebo různého formátu obrázku na základě vyhodnocení mediálního dotazu. (Michálek, 2015)

3.10.3 Formát SVG

Díky novým technologiím a stále se zvyšujícímu rozlišení obrazovek začíná být klasický bitmapový typ obrázků nedostačující. Řešením je vektorová grafika, konkrétně formát *SVG*[1].

Samotný formát existuje již od roku 2001, jeho rozvoji však bránil konkurenční formát *VML*, vyvíjený společností Microsoft. Vydání Internet Exploreru 9 však přineslo podporu právě i formátu *SVG*, a tak nic nebránilo tomu začít jej využívat. (Michálek, 2015)

Mezi hlavní výhody tohoto formátu patří možnost škálovatelnosti. To znamená, že obrázky lze následně rozšiřovat, vkládat do DOM struktury, editovat pomocí JavaScriptu, či kaskádových stylů. Obsah obrázků je také strojově čitelný, takže může být indexován vyhledávači. Ve skutečnosti je formát *SVG* jen *XML* dokument. Obrázky tedy jde otvírat a editovat v obyčejném textovém editoru. (Gasston, 2015)

Pro vkládání obrázků do webových prezentací lze využívat 2 různé přístupy:

- Vkládání zdrojového kódu obrázku přímo do stránky

```
    <embed src="img.svg"> definování obrázku </embed>
```

[1] Scalable Vector Graphics

nebo

```
<object src="img.svg"> definování obrázku </object>
```

- Odkazování na již hotový obrázek s koncovkou .svg

```
<img src="cesta-k-souboru.svg">
```

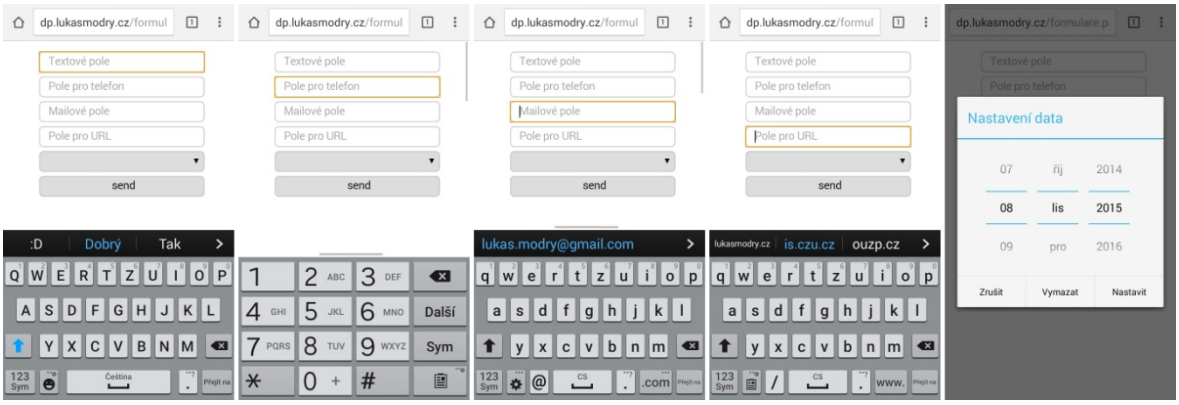
3.11 Nové prvky HTML5 pro tvorbu responsivního designu

3.11.1 Formuláře

Vytváření webových formulářů je obvykle velmi náročná a zdlouhavá práce, která obnáší značné množství psaní dodatečného kódu, napsaného jak v jazyce CSS, tak v JavaScriptu. Vývojáři se snaží své formuláře co nejvíce zefektivnit, zabezpečit a minimalizovat množství chybných dat, které se uživatelé snaží do jednotlivých polí vložit. (Castro, a další, 2012)

V této části práce budou popsány jen ty nejzákladnější formulářové prvky. Donedávna byla specifikace HTML omezena jen na pár prvků pro sběr dat od uživatele. Tím nejčastějším je klasické textové pole, se kterým se uživatelé setkávají prakticky u všech webových formulářů. Specifikace HTML5 však přináší nové formulářové typy, které práci značně zjednodušují. Hlavní myšlenkou těchto prvků je zanesení kontroly vložených dat přímo do specifikace HTML. Díky tomu není nutné využívat další skripty, které by vyhodnocovaly formu vložených dat a zabránily tak odeslání nevhodně vyplněného formuláře. (Lubbers, a další, 2011)

Nové prvky však přináší i zefektivnění práce na mobilních zařízeních jako jsou chytré telefony a tablety. Přestože zobrazení většiny formulářových prvků vypadá zcela totožně jako klasické textové pole, a nijak se bez podpory kaskádových stylů nemění jejich velikost a rozmístění, umožňují uživatelům snazší zadávání dat na virtuálních klávesnicích, které jejich zařízení nabízí. Pro jednotlivé typy elementů pro vkládání textů jsou u chytrých telefonů a tabletů navrženy různé rozložení kláves pro snazší přístup ke konkrétním znakům, jako jsou například @, / či &. (Hogan, 2012) Následné ukázky jsou zobrazeny na chytrém telefonu Samsung S4 Mini s operačním systémem Android 4.4.4 a výchozím webovým prohlížečem.



Obrázek 14 - Rozvržení klávesnice u polí pro vkládání textů v HTML5

Pro porovnání klasické textové pole pro vkládání textů:

```
<input type="text" name="text-field">
```

Toto textové pole poskytne uživateli klasické QWERTY rozložení klávesnice, na které je zvyklý například ze psaní SMS zpráv.

Dalším typem je pole pro vkládání telefonního čísla, které se obvykle používá při vyplňování registračních údajů na různých e-shopech apod.

```
<input type="tel" name="telephone-field">
```

To využívá klasickou číselnou klávesnici pro snazší zadávání telefonních čísel. Tím také zabrání uživatelům vkládat do pole pro telefonní číslo například textové řetězce. Podobně se zobrazí také klávesnice pro formulářový element typu *number*.

Třetím zmiňovaným element typu *email* určený pro vkládání emailových adres. Zápis je totožný jako u předchozích prvků, mění se pouze typ:

```
<input type="email" name="email-field">
```

Klávesnice, určená pro psaní emailových adres, je obohacena o znak zavináče a nejpoužívanější emailové koncovky *.com*.

Podobné rozložení má i čtvrtý zmiňovaný typ rozložení klávesnice pro vkládání URL adresy:

```
<input type="url" name="url-field">
```

V rozložení kláves je opět nepatrná změna - a to, že je zde přidáno lomítko pro oddělování složek a podsložek v URL adresách.

Poslední typ není úplně tak pole pro vkládání textových řetězců, ale element pro výběr data. Tento element uživateli zobrazí klasický proklikávací kalendář, kde si může

vybrat konkrétní den, měsíc a rok a vložit tuto hodnotu do formuláře. Alternativou k tomu je typ *datetime,* který umožňuje ještě výběr času. Tento typ má však podstatně horší podporu u prohlížečů.

```
<input type="date" name="date-field">
```

Využívání formulářových prvků jazyka HTML5 velmi usnadní a zefektivní práci s formuláři a přesněji definuje typ vložených dat. Další výhodou je jejich zpětná funkčnost v prohlížečích, které prvky nepodporují. Tyto elementy se zde zobrazí jako klasické textové pole bez rozšířených možností. (Castro, a další, 2012)

3.12 CSS frameworky

Tvorba webových stránek může být značně usnadněna využíváním CSS frameworků, v dnešní době už spíše frontend frameworků. (Michálek, 2013) Tyto nástroje poskytují možnost vytváření kvalitních, zajímavých a funkčních webových stránek masám uživatelů i s nižšími znalostmi jednotlivých technologií. Největší rozvoj zaznamenaly s příchodem webu pro mobilní zařízení. Těchto frameworků je nepřeberné množství, v této práci budou popsány jen dva nejpoužívanější – Bootstrap a Foundation. Frontend frameworky jsou často nazývány jako responsivní frameworky, to ale není zcela přesné označení, protože tyto nástroje řeší mnohem více, než jen přizpůsobitelnost webů. V praxi například řeší i práci frontendových designérů a kodérů. (Michálek, 2015)

3.12.1 Bootstrap

Tím nejznámějším je framework, vyvíjený společností Twitter, s názvem Bootstrap. Za jeho vznikem stojí Mark Otto a Jacob Thornton.

Bootstrap je postavený na jazyce HTML, sadě kaskádových stylů s množstvím mediálních dotazů, předdefinovaných vzhledů, často používaných prvků a dále na JavaScriptové knihovně JQuery, díky které je zajištěna funkčnost mnoha prvků. Bootstrap je volně stažitelná sada nástrojů pro tvorbu webových prezentací a aplikací.

Instalace Boostrapu do webu je velmi jednoduchá. Stačí do hlavičky webu zkopírovat zdrojové soubory a následně jen využívat jejich třídy a vlastnosti. Ty jsou k nalezení na oficiálních stránkách. Vedle toho však jde nástroj používat i modulově, tedy využívat jen určité funkce a nástroje. V tom případě není nutné implementovat celé soubory, ale jen jejich části. To zabezpečuje preprocesor LESS CSS. Hlavně v době

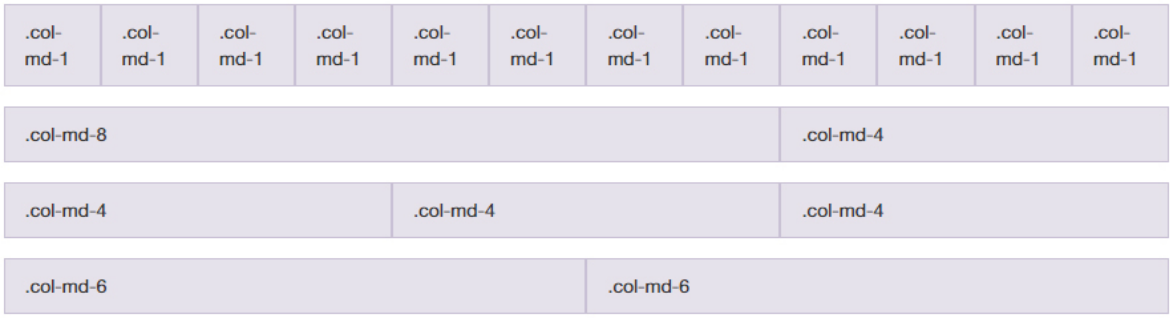
mobilního internetu a snaze o uspoření co nejvíce dat, je vhodné využívat právě modulární generování kódu Bootstrapu, což sníží jak datovou náročnost, tak i tu hardwarovou. Toto nastavení se odehrává v souboru *bootstrap.less*. (Vracovský, 2015)

Takový zápis by vypadal například následovně:

```
@import "bootstrap/less/variables.less";
@import "bootstrap/less/reset.less";
@import "bootstrap/less/type.less";
@import "bootstrap/less/forms.less";
```

Hlavní síla Boostrapu spočívá v přednastavených sadách kaskádových stylů. Samotná struktura je založena na 12 sloupcovém responsivního grid layoutu s možností volby 4 druhů zlomových bodů. Samotné řádky jsou následně oddělovány třídou `.row`. Rozměry jsou udávány v pixelech, což je hlavní měrná jednotka Bootstrapu:

- `.col-xs-` se zlomovým bodem menším jak 768px
- `.col-sm-` pro zlomové body větší nebo rovny 768px
- `.col-md-` pro zlomové body větší nebo rovny 992px
- `.col-lg-` pro zlomové body větší nebo rovny 1170px



Obrázek 15 - Sloupcové rozložení Bootstrapu, zdroj: (Bootstrap, 2015)

Další užitečnou funkcí je přednastavená typografie. V rámci stylů jsou připraveny styly pro nadpisy, různé označování textů, ale také pro zarovnání obsahu, různé transformace a podobně.

Bootstrap obsahuje mnoho různých přednastavených tříd pro většinu základních elementů jazyka HTML, jako jsou například tabulky, formuláře, odkazy, tlačítka a jiné. Ty však nejsou předmětem této diplomové práce a jejich škála je tak nepřeberná, že by pokryla obsah vlastní práce.

Jednou ze sad stylů pro podporu responsivního designu jsou třídy pro definování přizpůsobitelných obrázků. Konkrétně jde o třídu `.img-responsive`. Ta elementu, na kterém je použita, nastaví:

```
element {
  max-width: 100%;
  height: auto;
  display: block;
}
```

Dále je v Boostrapu pro responsivní design připravena sada stylů pro zobrazování a skrývání jednotlivých prvků na stránce. Třídy jsou znázorněny na následujícím obrázku.

	Extra small devices Phones (<768px)	**Small devices** Tablets (≥768px)	**Medium devices** Desktops (≥992px)	**Large devices** Desktops (≥1200px)
`.visible-xs-*`	Visible	Hidden	Hidden	Hidden
`.visible-sm-*`	Hidden	Visible	Hidden	Hidden
`.visible-md-*`	Hidden	Hidden	Visible	Hidden
`.visible-lg-*`	Hidden	Hidden	Hidden	Visible
`.hidden-xs`	Hidden	Visible	Visible	Visible
`.hidden-sm`	Visible	Hidden	Visible	Visible
`.hidden-md`	Visible	Visible	Hidden	Visible
`.hidden-lg`	Visible	Visible	Visible	Hidden

Obrázek 16 - Bootstrap - třídy pro zobrazování a skrývání prvků, zdroj: (Bootstrap, 2015)

Podobně lze pomocí 2 tříd rozlišit obsah, který se bude zobrazovat na displeji a obsahu, který bude poslán na výstup tiskárny.

- `.visible-print-block` – obsah, který se vytiskne
- `.hidden-print` – obsah, který se nevytiskne

Tyto sady stylů nejsou to jediné, co Bootstrap nabízí. To, co z něj dělá plnohodnotný frontendový framework, jsou komponenty. Mezi ty patří například sada ikon, jednoduchý responsivní carousel, či tvorba záložek pro třízení obsahu stránek (tzv. akordeon). (Michálek, 2015)

Otázkou však je, proč využívat předdefinované nastavení a komponenty frameworku. Jde především o usnadnění a zrychlení samotné práce na webu a využívání již hotových a ozkoušených technik. Je nutné si uvědomit, že přestože každý webový projekt

je možná unikátní, i tak se skládá z velmi podobných elementů. Velmi výstižně to shrnul David Heinemeier Hansson, který na přednášce na konferenci Future Of Web Applications řekl: *„Toto je sněhová vločka. Vaše aplikace není jedna z nich. Většina věcí, které většina lidí dělá, není nijak unikátní. Vaše potřeby nejsou nijak ‚unikátní'.“* (Hansson, 2006)

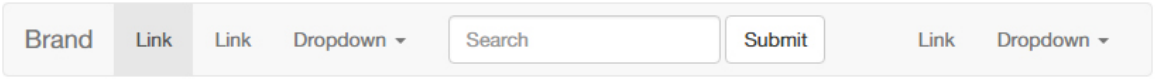
Podobně jako sad stylů, tak i komponent je zde značné množství. Zde budou popsány jen ty užitečné pro responsivní přístup k webu.

Jednou z nejužitečnějších komponent, která je k nalezení v tomto frameworku je *navbar*, neboli navigační panel pro konstrukci hlavního menu stránek. V samotném definování tohoto menu se nalézá podstatě více tříd, které definují různé další prvky. (Bootstrap, 2015)

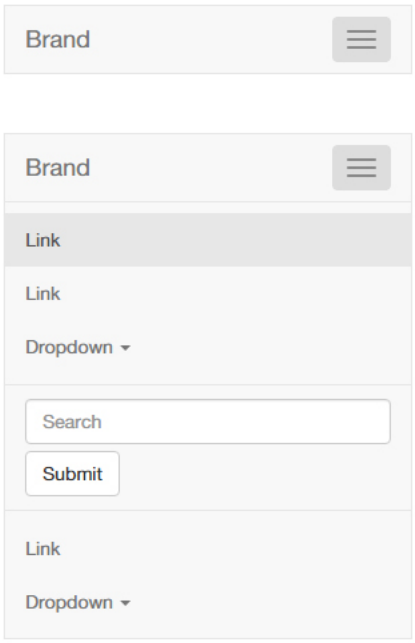
Základní specifikující navigační panely jsou:

- Obalového elementu `.navbar`
- Tlačítka pro zobrazení na malých rozlišeních `.navbar-toggle, .collapsed`
- Prostor pro logotyp `.navbar-brand`
- Seznam pro definování samotného menu `.nav, .navbar-nav`
- Prostor pro odkazy řazené vlevo `.navbar-left`
- Prostor pro odkazy řazené vpravo `.navbar-right`
- Dropdown odkazy `.dropdown-menu`

Výchozí navigační panel pak vypadá následovně:



Obrázek 17 - Bootstrap navigační panel – plné rozlišení, zdroj: (Bootstrap, 2015)



Obrázek 18 - Bootstrap navigační panel – mobilní rozlišení, zdroj: (Bootstrap, 2015)

Další velmi užitečnou komponentou je carousel pro vytváření responsivního image slideru. Ten funguje velmi obdobně, jako většina jiných JS pluginů pro vytváření sliderů s tím rozdílem, že je již součástí samotného Bootstrapu a funguje responsivně.

Jeho základní třídou je `.carousel`. Dále obsahuje ovládací prvky, znázorňující, který slide je momentálně aktivní: `.carousel-indicators`. A ve finále samotné slidy: `.carousel-inner`.

3.12.2 Foundation

Další, velmi používaný frameworke od společnosti Zurb, nese název Foundation. Obdobně, jako Bootstrap, je i tento framework založen na elementech HTML, předdefinovaných sadách kaskádových stylů podpořených JavaScriptem.

Instalace frameworku je prakticky totožná s instalací Bootstrapu. Vývojáři se mohou uchýlit buď k implementování celého balíku pomocí CDN[2] odkazů nebo si zvolit jen konkrétní moduly. Ty lze vygenerovat přímo na webu Zurb, případně je importovat prostřednictvím preprocesoru SASS.

[2] Content Delivery Network

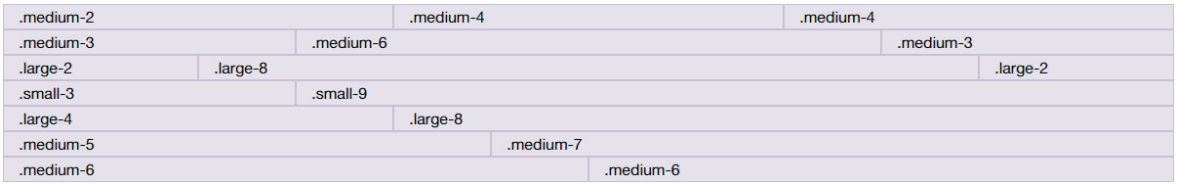
Foundation nabízí velmi podobný grid layout. Zásadním rozdílem je možnost volby z 3 velikostí oproti 4 v Boostrapu. Rovněž se však jedná o 12 sloupcovou mřížku, jejíž řádky jsou oddělovány třídou `.row`. Každá buňka pomyslné mřížky pak musí být obohacena třídou `.column`. (Zurb, 2016) Oproti Bootstrapu jsou však jednotky definovány relativně prostřednictvím jednotek *em*. Třídy pro určování sloupů vypadají následovně:

- `.small` - pro rozlišení obrazovky menší jak 40em
- `.medium` - pro rozlišení 40.063em až 60em
- `.large` - pro rozlišení 60.063em až 90em

Tyto třídy jsou rozšířeny ještě o 2 další, ty jsou však vhodné jen pro weby, které jsou připraveny pro rozlišení 2K a větší:

- `.xlarge` - pro 90.063em až 120em
- `.xxlarge` - pro větší jak 120.063em

Od těchto intervalů se odvíjejí veškeré zlomové body, spojené se třídami *small*, *medium* a *large*.



Obrázek 19 - Foundation grid layout, zdroj: (Zurb, 2016)

Ve Foundationu je i funkce na zobrazování či skrývání elementů v závislosti na rozlišení průhledu. Následné zobrazení či skrývání prvku je závislé na výše definovaných intervalech průhledu. Třídy pro zobrazování v závislosti na velikosti okna:

- `.show-for-medium` – pro zobrazení při *medium* a větší velikosti obrazovky
- `.show-for-large` – pro zobrazení na *large* a větších obrazovkách

Třídy pro skrývání v závislosti na velikosti okna:

- `.hide-for-medium` – pro skrývání na *medium* a větších obrazovkách
- `.hide-for-large` – pro skrývání na *large* a větších obrazovkách
- `.hide-for-small-only` – pro skrytí pouze na malé obrazovce
- `.hide-for-medium-only` – pro skrytí pouze na střední obrazovce
- `.hide-for-large-only` – pro skrytí pouze na velké obrazovce

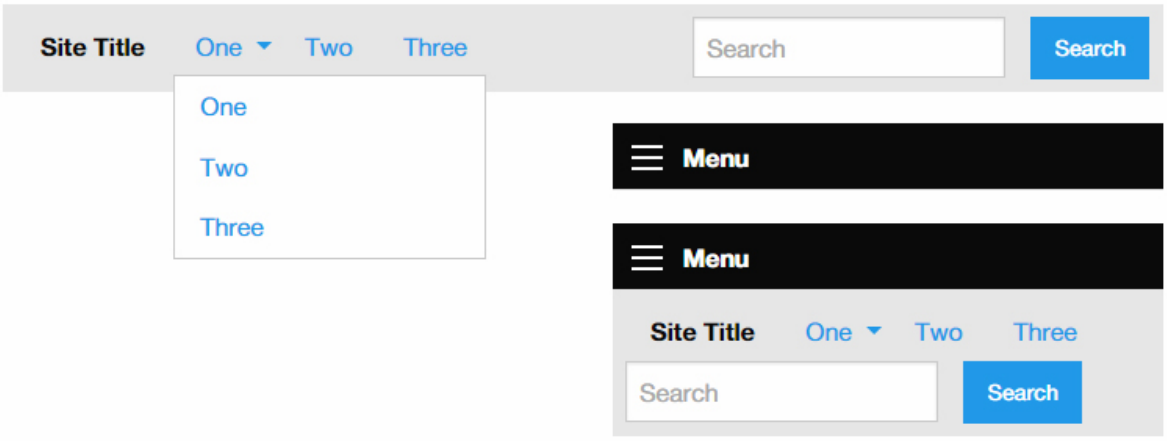
Třídy pro zobrazování v závislosti na režimu naklonění obrazovky:

- `.show-for-landscape` – pro zobrazení prvku při horizontálním naklonění obrazovky
- `.show-for-portrait` – pro zobrazení prvku při vertikálním naklonění obrazovky

Další komponentou je navigační panel, který se obdobně skládá z těchto prvků:

- Obalový element `.top-bar`
- Odkazy rovnané doleva `.top-bar-left`
- Odkazy rovnané doprava `.top-bar-right`
- Dropdown menu `.dropdown`, `.menu`
- Ikony pro zobrazení responsivního menu `.menu-icon`, `.title-bar-title`
- Loga společnosti `.menu-text`

Samotné výchozí menu z frameworku Foundation vypadá následovně:



Obrázek 20 - Navbar Foundation, nahoře v plném rozlišení, vpravo dole responsivní verze, zdroj: (Zurb, 2016)

4 Praktická část

V kapitole *„Teoretická východiska práce“* byly popsány veškeré klíčové možnosti pro tvorbu přizpůsobitelných webových stránek pro většinu typů zařízení. Na těchto teoretických znalostech jsou v praktické části realizovány webové prezentace za použití těch nejvhodnějších možností pro jednotlivé obsahové prvky. Samotné zdrojové kódy stránek jsou psány podle grafického návrhu za použití webového editoru PSPad 4.5.8, nikoliv *WYSIWYG*[3] aplikace.

4.1 Grafický návrh prezentace

Jednoduchá webová prezentace je navržena jako prostý blog pro sdílení článků na téma této diplomové práce. Stránka je realizována pouze jako statický web bez dodatečné administrace pro vytváření a úpravu obsahu, což není předmětem této práce.

Grafický návrh je realizován ve třech různých variantách pro rozlišení o šířce obsahu 480px, 992px a 1170px (při Full HD displeji). Podklady jsou navrženy v aplikaci Adobe Photoshop CS6.

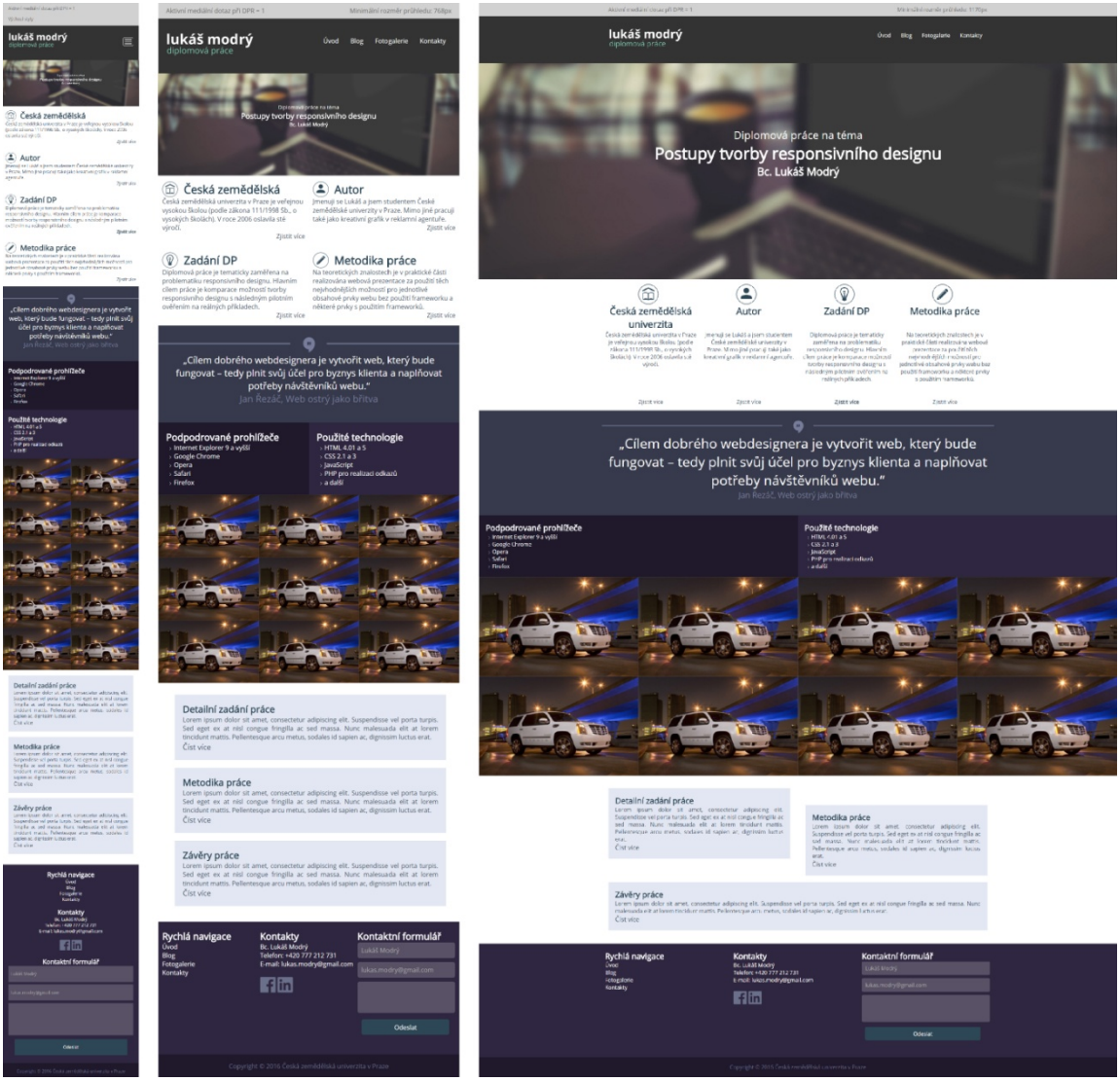
Úvodní strana obsahuje několik obvykle používaných částí, které jsou zároveň vhodné pro znázornění přístupů responsivního designu jak bez použití frameworků, tak s nimi. Těmi jsou:

- Zápatí webu s identifikací zlomového bodu a hodnotou DPR
- Navigační lišta s logem, jednotlivými odkazy a dropdown ovládacím panelem
- Pojízdný image slider s názvem webové prezentace a jménem autora
- 4 informativní boxy o obsahu webu, sloužící jako rozcestník
- Panel o 100% šířce s citátem souvisejícím s responsivním designem
- 2 sloupový panel s obsahem
- Vícesloupcová fotogalerie
- Sekce s různě rozloženými boxy se základními informacemi
- 3 sloupcová patička webu s rychlou navigací, základním kontaktem, kontaktním formulářem a dále autorskými právy

Návrhy pak vypadají následovně a slouží jako předloha k tvorbě samotné webové prezentace. Ta však není vytvářena 1 : 1 k návrhu, nýbrž tak, aby posloužila jako ukázkový

[3] What You See Is What You Get

příklad k tvorbě responsivního designu. Přesné stylování webu do daného vzhledu není předmětem této práce.



Obrázek 21 - grafické návrhy webu, zleva 480px, 992px, 1170px (1920px)

V následujících kapitolách jsou zhodnoceny a popsány nejvhodnější možnosti, jak dosáhnout požadovaného responsivního výsledku. Ty budou vybrány na základě testování a porovnání několika různých variant. Některé příklady jsou dále znázorněny i za pomoci komponent frontendových frameworků Bootstrap 3.3.6 a Foundation 6. Z důvodu nepřekročení stanoveného rozsahu této práce byly vybrány pouze dva nejpoužívanější frameworky mezi vývojáři webových stránek. Ukázkové weby jsou k dispozici na následujících odkazech:

- Webová prezentace bez použití frameworku: http://dp-clear.lukasmodry.cz/
- Framework Bootstrap 3.3.6: http://dp-bootstrap.lukasmodry.cz/
- Framework Foundation 6: http://dp-foundation.lukasmodry.cz/

Samotné ukázkové weby nejsou 100% kopií, nýbrž pouze ukázkou možností jednotlivých použitých variant realizace. V první jmenované variantě dále budou ukázány příklady, které nejsou součástmi komponent použitých frameworků.

4.2 Nastavení viewportu

Prvním krokem je volba vhodného průhledu. Cílem správně nastaveného *viewportu* je poskytnutí čitelného obsahu bez nutnosti horizontálního posouvání či přibližování obsahu stránek. Pomocí atributu *initial-scale* je možné určit základní velikost průhledu. V názorném příkladu lze použít různé nastavení na škále od 0.7 do 1.0. Při hodnotě menší jak 0.7 se obsah stává příliš malým pro prohlížení na mobilních zařízeních a vzniká tak nutnost manuálního přibližování. Naopak při větším zobrazení vzniká nutnost horizontálního posunu. Pro pohodlné čtení je v příkladu zvolena hodnota 1, tedy 100% velikost webové stránky.

Dalším parametrem, nad kterým se musí vývojáři zamyslet, je umožnění přiblížení samotného obsahu. To je vhodné například při potřebě zobrazit detailní náhled některých prvků, jako jsou fotografie, mapy a jiné podobné prvky. Dalším možným použitím zoomování je při potřebě používání malých ovládacích prvků, jako jsou například formulářová pole, či tlačítka na přepínání obrázkového slideru. To však pro uživatele není příliš pohodlné a stává se tak nežádoucím. V tomto případě lze nastavit minimální a maximální velikost samotného průhledu pomocí vlastností *minimum-scale* a *maximum-scale*.

Naopak pro prohlížení obyčejného webu, který slouží pouze ke sdělení prostého obsahu je toto přibližování nevhodné. Navržená webová prezentace je nastavena tak, že není nutné přibližovat jednotlivé elementy. Z toho důvodu je tato vlastnost zakázána pomocí atributu *user-scalablei* s hodnotou *0*.

Posledním použitým parametrem meta značky *viewport* je definování základní šířky obsahu, kde je opět na zvážení, jaký obsah se bude na stránkách zobrazovat.

Například v případě zobrazování některých infografik, kde je potřeba umožnit uživateli prohlížení obsahu v jednom řádku, je vhodné definovat tuto šířku na velikost daného obsahu. V případě, že není klíčové, v jakém momentě se prvky začnou zalomovat pod sebe, je nejvhodnější variantou nastavit šířku webové stránky na šířku samotného zobrazení. K tomu složí hodnota *device-width* u atributu *width*.

Samotné nastavení průhledu v použitých příkladech je tedy následovné:

```
<meta name="viewport" content="initial-scale=1, user-scalable=0, width=device-width">
```

Díky těmto vlastnostem se webová stránka na mobilním telefonu[4] načte čitelná v dostatečné velikosti bez nutnosti zbytečného posouvání obsahu.

Obrázek 22 - Výchozí zobrazení webu po nastavení viewportu

Nastavení meta tagu *viewport* je podporováno ve většině posledních verzí prohlížečů. Omezená podpora je u Internet Exploreru a jeho následovníka Edge, jež nepodporují atribut *vmax*, který však v ukázkovém příkladu není použit. Jediný z obvykle používaných prohlížečů, který tuto meta značku nepodporuje, je mobilní aplikace Opera Mini. Tu globálně využívá podle statistik na webu *CanIUse* téměř 5% uživatelů, na území

[4] Screenshot na telefonu Samsung Galaxy S4 Mini

ČR je to však pouhých 0,39%. (Deveria, 2016) Webové stránky se však v prohlížeči zobrazí čitelně, takže není potřeba se tímto nedostatkem více zaobírat.

Meta značka s definováním *viewportu* je jednou ze základních a nejvlivnějších vlastností v HTML kódu webové stránky pro responsivní zobrazení a vždy je nutné ji řádně nastavit a otestovat napříč jednotlivými zařízeními.

4.3 Rozvržení layoutu webu

Pro definování základní struktury musí vývojáři zvolit vhodné metody pro rozvržení obsahu celé webové prezentace. Je velmi užitečné se dopředu zamyslet nad tím, jak budou prvky na stránce uspořádány a jak se budou v závislosti na rozlišení obrazovky přeskupovat. K tomu slouží výše zmíněný grafický návrh. Při špatném rozvržení může být zdrojový kód zbytečně složitý a rozsáhlý, díky čemuž se mohou webové stránky pomaleji načítat. Špatně rozvržená struktura webu je také následně velmi složitě upravovatelná, což zpravidla navyšuje náklady na správu stránek samotných - především v momentě, kdy se k úpravě stránek dostane programátor, který nebyl u zrodu původní stránky. Způsob rozvržení obsahu je nejdůležitější fází při vývoji responsivních webových stránek. Ve většině případů vývojáři využívají mediální dotazy pro stanovení zlomových bodů.

Pro rozvržení obsahu webové stránky jsou využívány klasické obalové elementy *div* doplněné sémanticky nazvanými třídami. Alternativou jsou přímo sémantické značky HTML5.

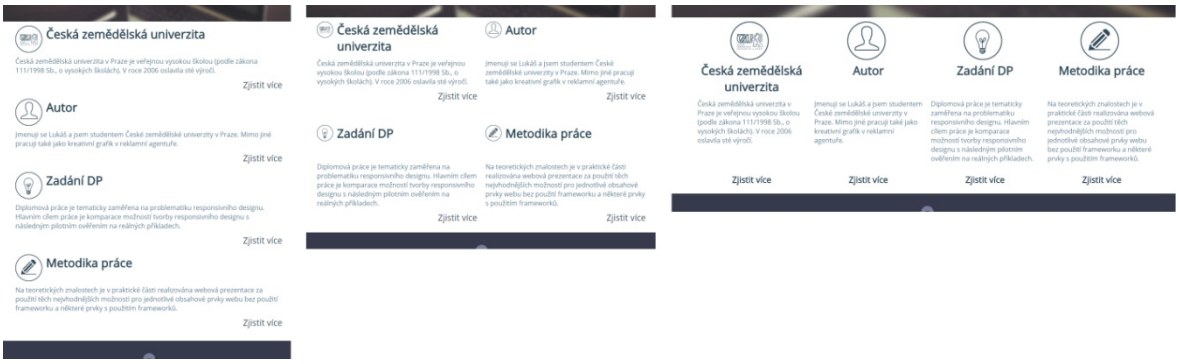
Při rozvrhování obsahu bez použití frameworku je nutné veškeré obalové elementy, jejich rozměry, zlomové body a další vlastnosti zobrazení nastavit pomocí kaskádových stylů. Při technice *Mobile First* je v první řadě vytvářen web, vhodný pro prohlížení na těch nejmenších zařízeních a ten je dále rozšiřován o další vlastnosti. S tím je spojena i volba jednotlivých zlomových bodů. Pro ukázku byly zvoleny 3 následující:

- `@media (min-width: 768px) { … }`
- `@media (min-width: 992px) { … }`
- `@media (min-width: 1170px) { … }`

Samotné ukázkové webové stránky mají tedy 4 různé předpisy kaskádových stylů pro určité elementy a k nim přiřazené třídy. Základní sada je pro veškeré typy zobrazení s menším průhledem jak 768 pixelů. Následně pak pro obrazovky mezi 768 až 991px,

992px až 1169px a pro největší obrazovky jsou připraveny poslední styly, načtené od 1170px. Při tvorbě responzivních webových stránek lze zvolit podstatně více zlomových bodů a jiných typů mediálních dotazů. Pro názornou ukázku v této práci je však tento počet a typ dostačující.

Následující ukázky jednotlivých přístupů budou popsány na rozvržení obsahu 4 základních boxů, které slouží jako rozcestník. Záměrem je poskládat tyto boxy v nejmenším rozlišení pod sebe a naopak při největším možném je umístit vedle sebe. Pro názornou ukázku možností je dále nastavena přechodová fáze, kdy se boxy zúží na polovinu šířky obalového elementu a vytvoří dva řádky po dvou sloupcích.



Obrázek 23 - Ukázka 4 sloupcového layoutu a rozvržení jeho obsahu

4.3.1 Klasický přístup k rozvržení webové stránky

Při rozvrhování responsivního obsahu za pomoci klasických postupů se vývojáři často uchylují k využívání třídy, která rozděluje jednotlivé řádky pomyslné mřížky pomocí vlastnosti *clear*. V příkladu tato třída nese totožný název `.clear`. Konkrétně pak:

```
.clear { clear: both; }
```

Tuto třídu lze použít na kterýkoliv element jazyka HTML, nejčastěji však na prázdný obalový element *div*. Případně lze třídu doplnit o další nastavení, jako je vypnutí okraje, výšky, odsazení a následně ji použít například u elementu *hr*.

Dále byla vytvořena třída, která bude omezovat obsah od okrajů zobrazení. Ta nese název `.content` a definuje šířku obsahu. V základu je třída nastavena na 100% šířky zobrazení, tedy plné pokrytí obrazovky od okraje k okraji. Při rozlišení větším jak 1170px se však omezí na tuto šířku a díky vlastnosti *margin* doplní rovnoměrně prázdný prostor na obě strany.

```
.content { width: 100%; }

@media (min-width: 1170px) {
  .content {
    width: 1170px;
    margin: 0 auto;
  }
}
```

Pro určení vícesloupcového layoutu je zapotřebí využívat u jednotlivých elementů vlastnost *float*. Pro vytvoření 4 sloupcového layoutu, který je použit jako ukázkový příklad, je tedy nutné vytvořit libovolnou třídu, která bude mít nastaveno obtékání obsahu zleva a bude určovat šířku jednotlivých boxů. V příkladu se tato třída jmenuje `.columns` a je nastavena na specifickou šířku na základě velikosti průhledu. Její použití na struktuře jednotlivých elementů je velmi jednoduché:

```
<div class="content">
  <div class="columns">...</div>
  <div class="columns">...</div>
  <div class="columns">...</div>
  <div class="columns">...</div>
  <div class="clear"></div>
</div>
```

Třída je v základu nastavena tak, aby vyplnila celou šířku zobrazení. Samotný prostor je omezen na velikost elementu s přiřazenou třídou `.content` a pro ošetření chyb při obtékání je pod boxy dále umístěn element *div* se třídou `.clear`. Na jednotlivých zlomových bodech jí je dále nastaveno zmíněné obtékání a omezení její velikosti.

```
.columns { width: 100%; }

@media (min-width: 768px) {
  .columns {
    width: 50%;
    float: left;
  }
}

@media (min-width: 992px) { .columns { width: 25%; } }
```

Z toho je patrné, že jednotlivé bloky se v závislosti na velikosti zobrazení zmenšují a postupně rovnají vedle sebe.

Dále je nutné definovat samotný obsah těchto boxů. V první řadě je pro správné zobrazení vhodné především nastavení shodné výšky pro jednotlivé nadpisy a odstavce textu v těchto boxech z důvodu, aby nedocházelo k nerovnoměrnému rozložení prvků. Responsivní návrh této webové stránky zahrnuje přesunutí ikony jednotlivých boxů z pozice před nadpisem do pozice nad ním. To je zajištěno opět změnou v obtékání elementů. Samotná ikona má dále nastavenu relativní velikost pro přizpůsobení zobrazení. Její ostrost je při velkém zvětšení zajištěna vektorovým formátem SVG, který je v této práci popsán dále. Ve zdrojovém kódu stránky jsou obrázek a nadpis nastaveny takto:

```
.columns h2 {
  overflow: hidden;
  height: 40px;
}

.columns img {
  float: left;
  width: 15%;
  max-width: 40px;
  height: auto;
  margin-right: 10px;
}
```

```
@media (min-width: 992px) {
  .columns h2 {
    text-align: center;
    height: 80px;
  }

  .columns img {
    float: none;
    width: 25%;
    max-width: 60px;
  }
}
```

Velký problém nastává v momentě, kdy je potřeba mezi jednotlivé navzájem se obtékající elementy vložit nějaký prostor jako mezeru. Ukázka této situace je realizována v kontaktech u rozvržení 3 sloupcového layoutu, kdy je potřeba jednotlivé boxy od sebe oddělit. Z matematického výpočtu je patrné, že tyto sloupce by měly mít šířku 33.3333%. V momentě, kdy však budou následně od sebe odděleny mezerou o velikosti 4%, vzniká problém s přetékáním obsahu. Po přepočtení je velikost celého obsahu 33.3333%+33.3333%+33.3333%+4%+4%+4%, tedy po zaokrouhlení 116%. Z toho je patrné, že jednotlivé sloupce musí být zmenšeny, v příkladu konkrétně na hodnotu 30% šířky obalového elementu, díky čemuž vzniká 10% prostoru, který zůstává k dispozici na rozdělení mezi tyto obsahové prvky. Při používání jedné třídy pro všechny 3 sloupce však vzniká mezera i za posledním z nich, která je nežádoucí a tu je potřeba následně vypnout.

```
@media (min-width: 992px) {
  .threecolumn {
    width: 30%;
    float: left;
    margin-right: 4%;
  }

  .threecolumn.last {
    margin-right: 0px;
    float: right;
  }
```

V ukázce byla použita další třída, která se nalézá právě na posledním elementu. Alternativní možností je využití pseudotřídy, která tento prvek bude identifikovat.

Podobným způsobem lze realizovat celé rozvržení webové stránky. Je pouze nutné pro jednotlivé sloupcové varianty vytvořit unikátní třídy a zajistit jejich chování napříč mediálními dotazy. Z toho je patrné, že soubor s kaskádovými styly pro jednotlivá rozlišení bude poměrně obsáhlý.

4.3.2 Rozvržení webové stránky pomocí Flexboxu

Další možností, jak rozvrhnout obsah webové prezentace, je používání nové hodnoty vlastnosti *display* s názvem *flex*, kterou do specifikace přináší třetí generace kaskádových stylů. Ta pracuje s přiděleným prostorem, který poměrově rozdělí dceřiným elementům. HTML struktura pro stejný příklad vypadá následovně:

```
<div class="flex-columns">
  <div class="flex-column"> ... </div>
  <div class="flex-column"> ... </div>
  <div class="flex-column"> ... </div>
  <div class="flex-column"> ... </div>
</div>
```

Třída `.flex-columns` funguje jako obalový element, je tedy alternativou ke třídě `.content` v předchozí ukázce. Tento element určuje, jak se bude jeho obsah řadit za sebe a jakým způsobem se bude zobrazovat.

```
.flex-columns {
  display: flex;
  flex-direction: column;
  flex-wrap: wrap;
  width: 100%;
}

@media (min-width: 768px) {
  .flex-columns {
    flex-direction: row;
  }
}

@media (min-width: 1170px) {
  .flex-columns {
    width: 1170px;
    margin: 0 auto;
  }
}
```

Třída je v základu nastavena tak, aby se její obsah rovnal do sloupce, tedy pod sebe. To je vhodné především pro malé displeje. K tomu slouží vlastnost *flex-flow*, jejíž hodnota se při zobrazení větším, jak 768px změní na řádkové a jednotlivé prvky obsahu se tak začnou skládat vedle sebe. Při překročení kapacity obalového elementu je dále díky vlastnosti *flex-wrap* zajištěno, že přetékající obsah se zarovná na další řádek.

Velmi podobně lze rozvrhnout i obsah jednotlivých boxů, který se bude v závislosti na obsahu přeskupovat následovně:

Obrázek 24 - Responsivní přeskupování hlavních boxů

Samotné přeskupení lze realizovat jednoduše za pomoci vlastnosti *flex-wrap* s hodnotou *wrap*. K samotnému rozmístění elementů je nutné zajistit překročení dostupného prostoru.

```
.flex-columns a img {
  order: 1;
  flex-basis: 40px;
}
.flex-columns a h2 {
  order: 2;
  flex: 75%;
}
```

Z toho je patrné, že ikona tvořená obrázkem a nadpis druhé úrovně společně pokryjí většinu dostupného prostoru. V okamžiku, kdy se zbývajících 25% prostoru, vyhraněného pro ikonu nadpisu dostane pod hranici 40px, jednotlivé elementy se přeskupí.

Výše zmíněný problém s umístěním prostoru mezi jednotlivé elementy lze dále vyřešit pomocí vlastnosti *justify-content* s hodnotou *space-between*, která rozděluje zbývající prostor rodičovského elementu mezi jeho potomky, přičemž na začátek a konec nepřidá nic. Tím se eliminuje potřeba definování dalších dodatečných vlastností pro konkrétní prvky v obalovém elementu a značně se díky tomu usnadňují výpočty rozměrů jednotlivých boxů.

Při aplikaci například v kontaktech webové stránky by tedy obalový element byl obohacen o vlastnost *justify-content*. Při velikosti jednotlivých dceřiných boxů 30% obsahového prostoru by mezi jednotlivými prvky vznikla mezera o velikosti 5%.

Flexbox je mocný nástroj především pro definování textového a dalšího obsahu umístěného uvnitř webu. Jeho využívání je možné ve většině dostupných webových prohlížečů. Zásadní problém však nastává s podporou u Internet Exploreru ve verzích 8 a 9, kde tato vlastnost není vůbec podporována. Tyto prohlížeče mají stále poměrně velké zastoupení na trhu. Pro verze IE, které vlastnost podporují, je dále nutné používat prefix `-ms-`.

4.3.3 Rozvržení webové stránky pomocí Grid layoutu

Další možností, která je v době psaní této práce pouze součástí návrhu, je hodnota *grid* pro vlastnost *display*. Ta funguje na rozvržení disponibilního prostoru do mřížky. Struktura samotných elementů je téměř totožná, mění se jen předpis kaskádových stylů.

```
<div class="grid-columns">
  <div class="grid-column-1 grid-column">...</div>
  <div class="grid-column-2 grid-column">...</div>
```

```
    <div class="grid-column-3 grid-column">...</div>
    <div class="grid-column-4 grid-column">...</div>
  </div>
```

Třída *grid-columns* definuje hranice obalového elementu, který je rozdělen do pomyslné mřížky pomocí následujících vlastností.

```
  .grid-columns {
    display: grid;
    grid-columns: 1fr;
    grid-rows: 1fr 1fr 1fr 1fr;
  }
  @media (min-width: 768px) {
    .grid-columns {
      grid-columns: 1fr 1fr;
      grid-rows: 1fr 1fr;
    }
  }
  @media (min-width: 992px) {
    .grid-columns {
      grid-columns: 1fr 1fr 1fr 1fr;
      grid-rows: 1fr;
    }
  }
```

Z tohoto kódu lze vyčíst, že se struktura samotné tabulky mění na základě velikosti průhledu prohlížeče. V základní verzi pro nejmenší rozlišení obrazovek je nastavena jako 1 sloupcová 4 řádková tabulka. Při překročení 768px se tabulka přeformuje do velikosti 2×2 a pro největší obrazovky se stává již 4 sloupcovou mřížkou o 1 řádku.

Jednotlivé boxy jsou následně do buněk umisťovány pomocí souřadnic jednotlivých míst a přeskupovány na základě aktuální podoby mřížky. Podobným způsobem je následně rozdělen i obsah jednotlivých rozcestníkových boxů. Grid layout je však určen spíše pro organizování textů. Práce s ním je v rámci složitější struktury obsahu poměrně náročná a chaotická.

Protože samotná vlastnost *display* s hodnotou *grid* je zatím stále jen součástí návrhu, její použití v praxi nepřipadá v úvahu. Samotná funkce je podporována pouze v nejnovějších verzích Internet Exploreru a jeho nástupce Microsoft Edge. I pro tyto prohlížeče je však nutné používat prefix `-ms-`.

4.3.4 Rozvržení layoutu pomocí Bootstrapu

Možností, jak si vývojáři mohou usnadnit práci při rozvrhování obsahu webové stránky, je používání frameworků. Tím nejpoužívanějším grid systémem je bezpochyby ten, který je předdefinovaný ve frameworku Bootstrap. V něm se využívá několika základních tříd, díky kterým lze určit rozvržení obsahu. Tyto třídy je vhodné přiřazovat elementu *div*, případně novým strukturálním značkám HTML5, jako jsou *header*, *section*, *nav*, apod.

Základní třídy pro definování obsahu použité v příkladu jsou `.containter` a `.container-fluid`. První jmenovaná omezuje základní šířku podle jednotlivých zlomových bodů, funguje tedy obdobně jako třída `.content` ve vlastním řešení. Zbývající prostor průhledu je rovnoměrně rozdělen na obě strany tohoto prvku. Vnitřní odsazení elementu je nastaveno na 15px od levého a pravého okraje. Tato třída je užitečná především pro textový a jiný obsah, který je třeba umístit na střed stránky s předdefinovanou šířkou.

Bootstrap využívá velmi podobné rozvržení zlomových bodů, které byly zmíněny při realizaci webu bez použití frameworku. Využívá tří základních zlomů, které jsou nastaveny na 768px, 992px a 1200px.

- Průhled menší jak 768 pixelů – šířka elementu *container* je 100%
- `@media (min-width: 768px)` – šířka elementu *container* je 750px
- `@media (min-width: 992px)` – šířka elementu *container* je 970px
- `@media (min-width: 1200px)` – šířka elementu *container* je 1170px

Druhá jmenovaná, `.container-fluid`, slouží k umístění obsahu vždy na celou šířku průhledu rovněž s 15px odsazením od levého a pravého okraje.

Jednotlivé řádky pomyslné mřížky jsou od sebe odděleny třídou `.row`. Ta funguje prakticky téměř totožně, jako třída `.clear`. Třída má v základu nastavenu vlastnost *clear* na hodnotu *both*, což zamezí nežádoucímu obtékání prvků. V manuálu frameworku je však doporučeno používat tuto třídu na obalový element jednotlivých buněk, nikoliv pouze jako oddělovač.

Buňky v mřížce layotu jsou v ukázkovém příkladu definovány pomocí velmi praktických přednastavených tříd. Ukázkový čtyř-sloupcový layout je definován pomocí 3 skupin stylů:

```
<div class="row main-boxes">
  <div class="container">
    <div class="col-lg-3 col-md-6 col-sm-12">...</div>
    <div class="col-lg-3 col-md-6 col-sm-12">...</div>
    <div class="col-lg-3 col-md-6 col-sm-12">...</div>
    <div class="col-lg-3 col-md-6 col-sm-12">...</div>
  </div>
</div>
```

V příkladu je využita skupina různě definovaných tříd pro vyplnění 12 sloupcové mřížky. Jejich název se skládá z 3 částí:

- `col` – určuje, že se jedná o sloupec
- `sm`, `md`, `lg` – určuje, při jakém rozlišení bude třída aplikována
- Hodnota `1-12`, v příkladu tedy hodnoty 3, 6 a 12, pro určení, kolik buněk pomyslné mřížky bude pokryto

Při nejmenším rozlišení jsou tedy prioritně načteny vlastnosti třídy `.col-sm-12`. Ta rozvrhne obsah boxu přes všech 12 sloupců, díky čemuž jsou tyto 4 boxy zarovnány pod sebe. Při středním rozlišení obrazovky je však využita třída `.col-md-6`. Právě hodnota 6 říká, že jednotlivé boxy pokryjí 6 buněk, tedy 2 boxy na řádek. Pro největší rozlišení je nastavena hodnota `.col-lg-3`, která srovná všechny boxy vedle sebe.

Nastavení rozměrů a obtékání těchto prvků je přednastaveno již v předpisu stylů samotného frameworku a není tak potřeba se jimi dále zabývat, pokud není požadavkem například změna jejich výchozího vnitřního odsazení. Ty jsou ve výchozím zobrazení nastaveny na 15px od levého i pravého okraje, což řeší i problém, který vznikl při realizaci boxů v kontaktech. Nevýhodou však je, že toto odsazení je nataveno na každý sloupec z obou stran. Obsah tedy bude odsazen o okraje celého obalového elementu a tudíž zúžen celkem o 30px, což může způsobit problém při snaze dodržet grafický návrh celé stránky.

4.3.5 Rozvržení layoutu pomocí Foundationu

Obdobný systém nabízí také framework Foundation. I ten využívá předdefinované třídy, vhodné k používání na obalových elementech. Tyto třídy jsou však trochu jinak

přednastaveny. Stejně jako v Bootstrapu, i zde existuje třída `.row`, která odděluje jednotlivé řádky pomyslné mřížky. Součástí této třídy je však i nastavení maximální šířky elementu, na který je použita. Z toho důvodu není možné tak snadno nastavit obsah na celou šířku prohlížeče. Rozdíl v nastavení jednotlivých tříd je znázorněn v následující ukázce:

V Bootstrapu je tato třída nastavena takto:

```
.row {
  margin-left: -15px;
  margin-right: -15px;
  dislpay: table;
  clear: both;
  box-sizing: border-box;
}
```

Naopak ve Foundationu vypadá třída následovně:

```
.row {
  max-width: 75rem;
  margin-left: auto;
  margin-right: auto;
  display: table;
  box-sizing: inherit;
  clear: both;
}
```

Z toho vyplývá, že Foundation se snaží řešit problém s omezováním obsahu od okrajů stránky a zároveň oddělování jednotlivých bloků obsahu v rámci obtékání pomocí jedné třídy místo dvou, jako je tomu v konkurenčním frameworku. Alternativa pro využití celé šířky obalového elementu zde není k dispozici a je třeba ji vytvořit. V použitém příkladu se jedná o třídu `.full-width`, která je nastavena následovně:

```
.full-width {  max-width: 100%;  }
```

Pro určení sloupců mřížky je ve Foundationu k dispozici rovněž 12 sloupcový layout. Pomocí 3 typů tříd lze definovat jednotlivé hraniční rozměry pro zalomení obsahu pod sebe. Samotné obtékání jednotlivých prvků je zajištěno pomocí třídy `.column`, která musí být doplněna ke každé buňce mřížky.

```
.column {
  padding-left: .625rem;
  padding-right: .625rem;
  width: 100%;
  float: left;
}
```

Tyto třídy jsou rovněž názorně ukázány na rozvržení obsahu do 4 sloupcového layoutu u rozcestníkových boxů.

```
<div class="row main-boxes">
  <div class="large-3 medium-6 small-12 column">...</div>
  <div class="large-3 medium-6 small-12 column">...</div>
  <div class="large-3 medium-6 small-12 column">...</div>
  <div class="large-3 medium-6 small-12 column">...</div>
</div>
```

Z příkladu je patrné, že struktura je velmi podobná jako v Boostrapu. Podobné je i chování jednotlivých boxů v závislosti na rozlišení průhledu. Zlomové body však nejsou určeny pomocí přesných rozměrů zadaných v pixelech, jako je tomu v předchozích 2 ukázkách, ale pomocí jednotek *em*, které pracují s relativním rozměrem. Díky tomu je kontrola nad obsahem poněkud ztížena.

4.3.6 Doporučení k volbě grid systému

Znázorněné přístupy fungují na velmi podobném principu vytváření zlomových bodů a pozicování jednotlivých elementů kolem sebe. Všechny metody jsou založeny na používání mediálních dotazů. Ty jsou v době psaní této práce podporovány ve všech obvykle používaných webových prohlížečích, s výjimkou Internet Exploreru 8. Ten podle webu *Can I Use* využívá 1.18% uživatelů, což je v současné době již zanedbatelná část. (Deveria, 2016) Vždy je však nutné zvážit reálné uživatele konkrétní webové stránky a podle nich přizpůsobit podporu jednotlivých vlastností, případně vytvořit alternativní způsob zobrazování. Tím při nepodpoře mediálních dotazů v těchto příkladech bude nejmenší vzhled webu, tedy výchozí předpis stylů.

Při používání zastaralých metod s využíváním vlastnosti *float* je možné se zcela přizpůsobit potřebám jakéhokoliv grafického návrhu i podpory v prohlížečích bez nutnosti doplňování zdrojového kódu o prefixy jednotlivých renderovacích jáder. Zásadní problém však spočívá v náročnosti vytváření složitých struktur a jejich chování napříč jednotlivými

rozlišeními obrazovek. Zpravidla narůstá potřeba zvýšené kontroly a testování celého webu. Dalším nedostatkem při používání zastaralého přístupu je nutnost stálého přepisování pozičních ukazatelů, parametrů pro odsazení obsahu a celkově minimální automatizace samotného způsobu zobrazení. Je tedy nutné definovat každý element na stránce pro každý zlomový bod a klást větší důraz na testování prvků s relativní velikostí.

Tento problém z valné většiny řeší metoda s pomocí hodnoty zvané *flexbox*, která je součástí specifikace CSS3. Přestože se na první pohled zdá práce s ní poměrně složitá, při jejím správném nastavení je velmi užitečným nástrojem pro organizování obsahu na webové stránce. Díky širokému spektru použití je vhodná jak pro členění obsahu jednotlivých podstránek, tak se dá efektivně využívat i jako alternativa k obtékání při formování celé struktury webové stránky. Flexbox však v Internet Exploreru 8 a 9 není vůbec podporován a v novějších verzích vykazuje značné množství chyb i s používáním prefixů. Pro základní jednoduché rozložení například článků se však dá využívat již nyní. V případě složitějších struktur je však jeho používání zatíženo nutností testování.

Přesto se však dá s jistotou tvrdit, že v momentě, kdy vydavatelé webových prohlížečů zavedou do svých aplikací úplnou podporu *flexboxu*, respektive kdy uživatelé přestanou využívat zastaralé webové prohlížeče, stane se tato vlastnost při tvorbě webových prezentací jednou z klíčových.

Oproti tomu nová vlastnost, nesoucí název *grid layout*, se kvůli nedostatečně podpoře v prohlížečích vůbec nedoporučuje využívat. Pro členění obsahu celé stránky je to i poměrně nevhodný nástroj kvůli složitosti jeho zápisu. *Grid layout* jako takový se hodí spíše pro organizování textových řetězců v článcích a jiných obsáhlých textech. Pro rozvrhování složitějších struktur, které se budou v závislosti na velikosti průhledu měnit, je nevhodný.

Využívání jednotlivých frameworků velmi usnadňuje práci při rozvržení samotného rozložení celého webu. Hlavní problém nastává při unikátně složitých strukturách pro rozvržení vzhledu webové stránky. Těmi mohou být například asymetrické rozložení prvků, specifikování vlastních zlomových bodů, či vytváření zcela atypické webové prezentace. Vesměs se dá říct, že se jedná o webové prezentace, které nekorespondují s předdefinovaným mřížkovým layoutem. V těchto případech je vždy vhodné zvolit vlastní tvorbu kaskádových stylů pro rozvržení obsahu, což umožní úplně přizpůsobit se

požadavkům grafického návrhu. Vhodnou variantou může u většiny typů webových prezentací být kombinace frameworkových tříd, které budou dále doplněny vlastními třídami pro vyřešení částí webu, jež jsou pro předdefinované styly nevhodné. Tato alternativa je jednou z nejefektivnějších metod při tvorbě webových stránek.

Z tohoto postupu vychází i následné upravování předdefinovaných stylů ve frameworcích. Ty nemusí vždy kolidovat s grafickým návrhem webu a záměrem webdesignéra. Při vytváření klasických webových stránek nebo při realizaci webu, který je přímo navržen pro použití těchto frameworků, není problém uchýlit se k této variantě, která výrazně šetří čas při vytváření kaskádových stylů. Často je však potřeba doplnit sadu dalších doplňujících vlastností, které přizpůsobí web grafickému návrhu a požadavkům klienta. Obvykle tedy nastává situace, kdy již existující styly je nutné přepsat stejnými vlastnostmi - pouze s jinou hodnotou. Takový postup je přesto obvykle rychlejší, než definování kompletních kaskádových stylů bez frameworků.

V rámci rozložení obsahu se vhodněji chovají elementy nastavené třídami frameworku Bootstrap, které umožňují lepší kontrolu nad obsahem díky přesně stanoveným zlomovým bodům. Pro zcela čistý zdrojový kód je však nejlepší variantou vlastní realizace bez použití frameworku. Ta však přináší značná úskalí v rámci testování funkčnosti webových stránek.

4.4 Práce s mediálními dotazy

Předchozí kapitola využívala pouze nejzákladnější možný mediální dotaz, který je dostačující pro rozvržení základní struktury webové stránky pro ta nejobvyklejší zařízení. Mediální dotazy však poskytují podstatně více možností.

Předně je nutné zvolit metodu pro jejich vkládání do webové stránky. Specifikace nabízí 2 základní možnosti:

- Vkládání mediálního dotazu do hlavičky webové stránky
- Vkládání přímo do souboru kaskádových stylů

První jmenovaná možnost je vhodná především pro weby, které se na různých zařízeních budou chovat zcela odlišně. Výhodou této varianty je stažení jen konkrétního předpisu kaskádových stylů pro daný mediální dotaz. To může uživateli šetřit data při používání mobilních zařízení. Takové soubory mají jen pár kilobajtů.

Zásadní nevýhodou při tomto přístupu je změna velikosti okna. V případě, že budou například definovány různé kaskádové styly pro aktuální orientaci zařízení pro variantu *portrait* a *landscape*, může dojít k chybě zobrazení při otočení zařízení do druhé polohy. V takové situaci musí prohlížeč stáhnout další sadu stylů. Názorným příkladem nevhodnosti tohoto použití je uvedení situace uživatele, prohlížejícího si webové stránky v podzemní dráze. Tento uživatel si ve stanici načte online článek a po opuštění soupravy prostor s dostupným signálem se rozhodne přečíst si článek na displeji otočeném na šířku. V tu chvíli však zařízení nemá možnost připojení se k internetu a nastává problém se zobrazením, protože prohlížeč nemá k dispozici alternativní sadu kaskádových stylů. Z toho důvodu je vždy vhodné stáhnout veškeré možné varianty kaskádových stylů.

Další nevýhodou je nutnost definování všech možných nastavení pro jednotlivá rozlišení a varianty mediálního dotazu. V použitém přístupu *Mobile First* jsou nastaveny základní styly, které jsou následně rozšiřovány. To v praxi znamená, že například velikost a rodina fontů jsou nastaveny jen v základním rozlišení a následně jsou dále doplňovány dalšími vlastnostmi. Takový postup by však při používání jednotlivých souborů nebyl možný, respektive by musel být vytvořen další globální soubor s výchozími nastaveními pro jednotlivé prvky, které by byly následně doplňovány dalšími předpisy z jiných souborů.

Při tomto postupu je vývojář nucen neustálého přepínání mezi jednotlivými soubory stylů, které musí být následně kompatibilní. To znamená, že musí některé kroky definovat vícekrát. V ukázkovém webu tato metoda není použita, vypadala by však následovně:

```
<link href="style-default.css" media="all">
<link href="style-768.css" media="(min-width: 768px)">
<link href="style-992.css" media="(min-width: 992px)">
<link href="style-1170.css" media="(min-width: 1170px)">
```

Alternativou je zapisování mediálních dotazů přímo do souboru kaskádových stylů. Tato metoda je v mnoha ohledech vhodnější. Za prvé jsou veškeré styly staženy najednou a jsou tedy vždy a okamžitě k dispozici uživateli. Dále vývojáři ušetří kontrolu nad samotným předpisem z důvodu využívání pouze jednoho souboru stylů. Pokud tedy není potřeba definování stylů v extra souborech, je vždy vhodnější varianta definování pouze jednoho souboru s kaskádovými styly, který bude obsahovat veškeré mediální dotazy.

4.4.1 Používání mediálních typů

Využívání mediálních typů, tedy dotazů na typ zařízení, na kterém se bude webová prezentace zobrazovat, je značně omezeno podporou jednotlivých možností na většině zařízení. V praxi se však většina vývojářů spokojí se základní výchozí hodnotou, tedy *all*, případně s její alternativou v podobě *screen*. V ukázkové webové prezentaci je u všech mediálních dotazů použita právě výchozí varianta, respektive není vyplněna žádná hodnota a prohlížeč si tedy automaticky doplní hodnotu výchozí.

Přesto existuje podstatně více mediálních typů. V této práci však budou popsány jen ty nejčastější, se kterými se uživatelé mohou setkat v každodenním životě. Jedná se o dva základní mediální typy, které mají zásadní vliv na tvorbu responsivních webových stránek.

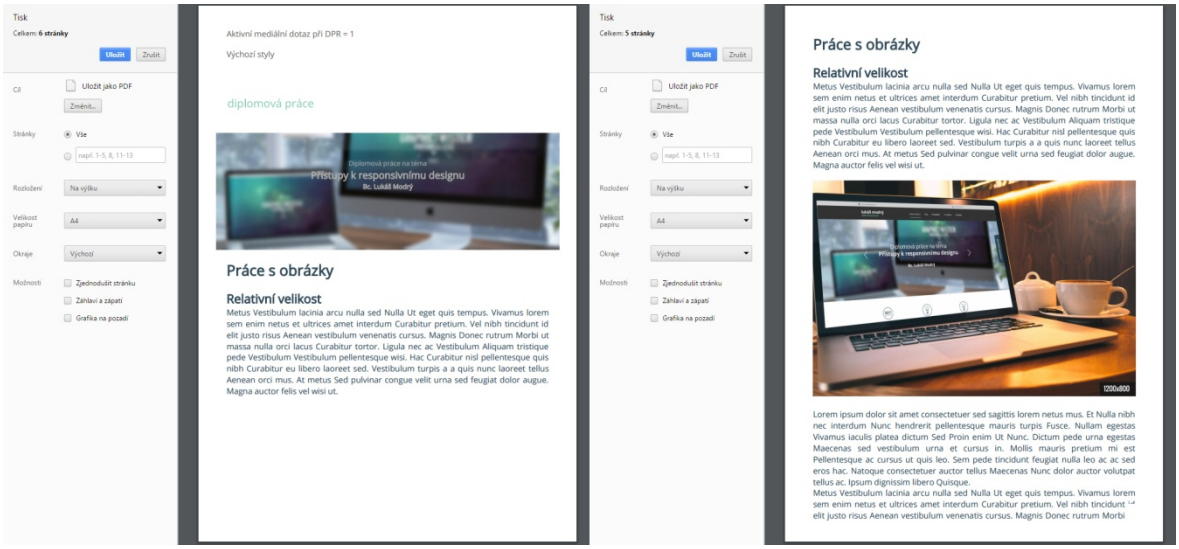
Tím prvním je *handheld*. Ten je určený pro zařízení, která uživatelé mají v ruce. Tento mediální typ je vhodný především pro zařízení s dotykovou obrazovkou, kde jsou jednotlivé prvky ovládány pomocí prstů. Uživatel se však často může setkat s tím, že jeho prsty jsou pro dané zařízení příliš velké. Tento problém nastává také v případě levnějších smartphonů a starších tabletů, které disponují horší dotykovou vrstvou. Díky mediálnímu typu *handheld* však lze konkrétní prvky na stránce například zvětšit, což usnadní jejich ovládání. Tento mediální typ však v době psaní této práce není téměř nikde podporován a uživatelům se tak načtou pouze výchozí kaskádové styly. Je tedy vhodnější uchýlit se k alternativě s mediálními výrazy, konkrétně na parametry šířky zobrazení. Ty budou v této práci ještě popsány.

Druhým je *print*, který slouží k definování kaskádových stylů, jež budou aplikované při tisku webové stránky. To je velmi užitečná věc například u různých blogů, či jiných typů webů, které mohou být vhodné k tisku. Autor webové stránky tak může zjednodušit barevnost webové prezentace, omezit její obsah o k tisku nepotřebné prvky, případně zjednodušit celou strukturu stránek samotných.

V ukázkové webové prezentaci je tento mediální dotaz použit k vypnutí nepotřebných částí webové stránky, která se nehodí k tisku a se samotným obsahem nemá příliš společného. Je tedy vypnut horní panel, hlavní navigace a obrázkový slider. Dále je při zasílání na tisk vypnuta patička webu. Takto použitý mediální dotaz se selektorem *print* vypadá velmi jednoduše, jak ilustruje následující příklad:

```
@media print {
  .top, .mainmenu, .rslides, footer {
    display: none;
  }
}
```

I takto jednoduchý zápis dotazu může mít zásadní vliv na vzhled a efektivnost tisku. Názorná ukázka na následujícím obrázku je při zobrazení webové stránky před a po použití tohoto kusu kódu při snaze vytisknout článek na ukázkové webové stránce.



Obrázek 25 - použití mediálního typu *print, vlevo varianta bez mediálního dotazu, vpravo s ním*

Z příkladu je patrné, že vývojář webové stránky může určovat prioritu konkrétnímu obsahu, který na webové stránce zanechá při zaslání na tisk, a tudíž uživatel nebude muset plýtvat barvou na ostatních, nepotřebných částech, které by stejně vyhodil. Tento mediální typ je podporován ve všech dnes dostupných webových prohlížečích a je tedy vhodné minimálně základní použití webu pro tisk používat, především v momentě, kdy je očekáváno, že by uživatelé mohli mít zájem o tisk obsahu daných webových stránek.

Speciální mediální typy, jako je například *braille*, *embossed*, *speech* a podobně, jsou spíše doplňkovou možností mediálních dotazů, se kterou se málokdy můžou návštěvníci webových stránek setkat. Pokud se nejedná o nějaký speciální web, kde jsou tato nastavení opravdu nutné, není potřeba je definovat. Zpravidla se jedná o zařízení, jejichž chování v rámci zobrazení se nijak nemění v závislosti na chování uživatele. Je tedy vhodné je umisťovat do vlastních souborů, aby nedocházelo k jejich zbytečnému stažení na ostatních zařízeních. Jejich podpora však není dostatečná a jejich použití je ovlivněno

mnoha dalšími komplikacemi. Například u mediálního typu *tv*, který je určen pro televize, je alternativa v podobě *screen* zcela dostačující. Význam by vznikl pouze v momentě, že by bylo potřeba zobrazovat jiný obsah či vzhled na televizní obrazovce a na jiném zařízení. Těmi mohou být například různé ovládací prvky apod. Při použití těchto mediálních typů nedojde k narušení webové stránky na ostatních zařízeních a není tak problém je definovat.

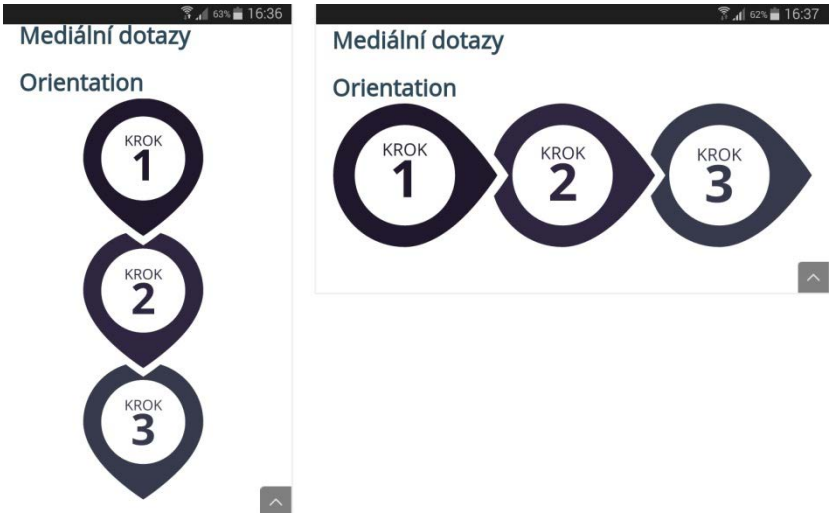
4.4.2 Využívání výrazů

Mediální typy dále mohou být obohaceny o další řadu specifikací v podobě mediálních výrazů, tedy vlastností daného konkrétního zařízení. Ty jsou klíčové pro definování jednotlivých zlomových bodů a jsou tedy neoddělitelnou součástí responsivního webu. V ukázkové webové prezentaci je nejpoužívanějším mediálním výrazem *min-width*, tedy dotaz na minimální šířku průhledu. Tento výraz je i podstatou přístupu *Mobile first*. Valná většina ukázkové webové stránky je realizována právě pomocí tohoto výrazu.

```
@media (min-width: 768px) { … }
@media (min-width: 998px) { … }
@media (min-width: 1170px) { … }
```

Na minimální šířce zobrazení jsou postaveny i oba porovnávané frameworky. Opačnou alternativou je *max-width*, tedy maximální šířka zobrazení. Ta je podstatou přístupu *Desktop first*. Velmi podobně se chovají i výrazy *device-width* a *device-height*, které pracují nikoliv s velikostí průhledu prohlížeče, ale s velikostí samotné obrazovky zařízení. To však v praxi není moc užitečné a využívá se pouze okrajově. Především proto, že u různých zařízení, která nepracují s okny, je parametr šířky, respektive výšky okna, shodný s hodnotou šířky a výšky samotného zařízení. Naopak u operačních systémů, které s okny pracovat umí, je nutné počítat s možností změny velikosti okna.

Užitečnou možností je sledování otočení samotného zařízení mezi polohami *portrait* a *landscape*. Ve skutečnosti se však nejedná o hodnotu, kterou by zasílalo zařízení na základě vyhodnocení svého gyroskopu, ale o poměr stran zobrazení. V momentě, kdy je šířka větší než výška, aplikují se styly definované pro *landscape* a naopak. Tento výraz byl použit na infografice v článku. Cílem je umožnit uživateli prohlížet si daný graf tak, aby to pro něj bylo pohodlné. V klasické poloze mobilního telefonu tedy bude uživateli nabídnuta varianta s horizontální orientací obrázku. Při otočení zařízení se však zobrazí horizontální obrázek.



Obrázek 26 - mediální výraz *orientation*

Při zápisu to pak vypadá následovně:

```
@media (orientation: landscape) {
  .orientace {
    width: 100%;
    background-image: url('../img/infografika-horizontalni.svg');
  }
}

@media (orientation: portrait) {
  .orientace {
    width: 50%;
    background-image: url('../img/infografika-vertikalni.svg');
  }
}
```

Mediální výraz je bez problémů podporovaný ve všech prohlížečích. Je tedy vhodné jej používat. Jeho aplikace na celé webové stránky by však byla poněkud složitá a zbytečná. Stačí se omezit jen na ty elementy, které je nutné změnit v rámci poměru stran prohlížeče. Nejvhodnější je tedy použití například u infografik, grafů a jiného podobného obsahu.

Pro volbu vhodných mediálních typů a výrazů je nutné se zamyslet nad cílovou skupinou uživatelů dané webové prezentace a zvážit jejich možné chování. Například u portálu poskytujícího videa je bezpředmětné počítat s tiskovou verzí webové stránky. Naopak u různých tiskových zpráv, návodů, nebo třeba receptů se naopak tato možnost

očekává. Je nutné definovat vhodné kaskádové styly pro veškeré možné alternativy zobrazení konkrétní webové stránky.

4.5 Navigační panel

Častým problémem, se kterým se vývojáři musejí při vytváření webových prezentací potýkat, je vytváření responsivního navigačního panelu. Ten slouží k procházení obsahu napříč webovými stránkami. Standardem je, že tyto panely se při malém rozlišení obrazovky omezí na pouhé tlačítko, které po kliknutí zobrazí jednotlivé odkazy.

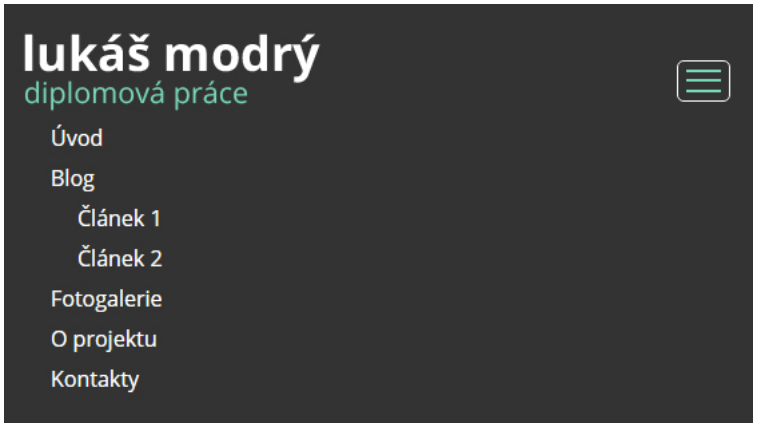
4.5.1 Navigace bez frameworku

Při realizaci hlavního menu webu bez využívání frameworků byla využita velmi jednoduchá struktura, která vypadá následovně:

```
<nav class="mainmenu">
  <div class="content">
    <a href="index.php" title="Diplomová práce">Brand webu</a>
    <ul id="menu">
      jednotlivé položky menu
    </ul>
    <span onclick="menuButton('menu'); return(false);"
      class="button"><hr><hr><hr></span>
  </div>
</nav>
```

Obalový element *nav* nastavuje základní vzhled hlavního panelu. Definuje jeho výšku pomocí vlastnosti *min-height*. Ta určuje jeho minimální rozměr, který se však může v závislosti na obsahu zvětšovat. Dále jeho barvu pozadí, šířku a jednotlivé odsazení od okrajů.

Jednotlivé odkazy jsou vytvořeny standardně pomocí nečíslovaného seznamu. V minimálním zobrazení je tento seznam skryt a místo něj se zobrazuje tlačítko pro jeho zobrazení.



Obrázek 27 - Responsivní navigační panel bez použití frameworku

Při nejmenším možném průhledu se místo hlavního navigačního panelu načte pouze tlačítko pro jeho zobrazení. Tuto funkci zajišťuje JavaScript. V příkladu se jedná konkrétně o funkci *menuButton*, která se zavolá po kliknutí na zmíněné tlačítko.

```
<script type="text/javascript">
  function menuButton (menu){
    var co;
    co = document.getElementById(menu);
    if (co.style.display == 'block'){ co.style.display = 'none';
    } else { co.style.display = 'block'; }
  }
</script>
```

Ta jednoduše zkontroluje hodnotu vlastnosti *display* u seznamu odkazů s identifikátorem `#menu` a na jejím základě ji změní na opačnou hodnotu. Tento skript lze obohatit o značné množství dalších funkcionalit, jako jsou různé efekty, způsoby zobrazení, a podobně. Při překročení velikosti průhledu 768px je toto tlačítko skryto a nahrazeno plnohodnotným navigačním panelem pro vyšší rozlišení. V zápise kaskádových stylů to v základu vypadá následovně:

```
.mainmenu ul { display: none; }

@media (min-width: 768px) {
  .mainmenu ul {
    display: block;
    float: right;
  }
}
```

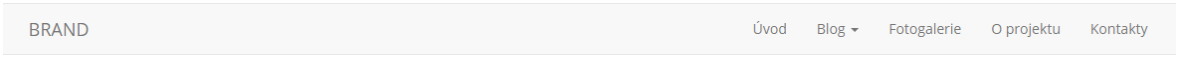
Tyto vlastnosti jsou následně doplněny o značné množství dalších stylů, které zajišťují vzhled jednotlivých položek. Zásadní změna se týká především druhé úrovně navigačního panelu. V ukázce je vypsáno pouze to, co se týká responsivního zobrazení:

```
.mainmenu ul { display: none; }

.mainmenu ul ul { display: block; }

@media (min-width: 768px) {
  .mainmenu ul ul {
    display: none;
  }

  .mainmenu ul li:hover ul {
    display: block;
  }
}
```

Z výše uvedeného příkladu vyplývá, že při malém rozlišení budou zobrazeny všechny položky navigace, a při větším rozlišení se budou zobrazovat po najetí myši. Tento postup je tím nejzákladnějším možným, v praxi je většinou potřeba definovat responsivní navigační panel blíže konkrétnímu grafickému návrhu a dále také jeho funkcionalitě, což je zpravidla spojeno s využíváním hotových JavaScriptových knihoven zajišťujících tyto efekty.

4.5.2 Navigační panel pomocí Bootstrapu

Při využívání frameworku Boostrap je pro vytváření navigačního panelu předpřipravena jasně definovaná struktura, která je doplněna hotovými kaskádovými styly i skriptem, zajišťujícím funkcionalitu samotného panelu. K nalezení je pod názvem *Navbar*. Při implementaci této struktury na seznam odkazů zvolený v ukázkovém příkladu vypadá panel následovně:



Obrázek 28 - Navigační panel v Bootstrapu bez dodatečných stylů

Pro dosažení požadovaného vzhledu je však nutné řadu hotových stylů ve frameworku nahradit jinými, případně je vypnout. Největší komplikací je přetvoření celého pozadí tohoto panelu.

Hlavní třída, určující pozadí a okraje panelu má výchozí nastavení takovéto:

```
.navbar-default {
  background-color: #f8f8f8;
  border-color: #e7e7e7;
}
```

V rámci ukázky je nutné tento okraj vypnout a změnit hodnotu *background-color* na *#333*. Tuto barvu je však nutné nastavit na další elementy, které mají ve výchozím nastavení pozadí buď vypnuté, nebo jinak barevné. Těmito třídami jsou:

- `.navbar-default .navbar-nav li a`
- `.navbar-default .navbar-nav li a:hover`
- `.navbar-default .navbar-nav li a:focus`

Samotným odkazům je následně vedle změny pozadí nutné změnit i barvu písma.

```
.navbar-default .navbar-nav .open a, .navbar-default .navbar-nav
.open a:focus, .navbar-default .navbar-nav .open a:hover
{
  color: white;
  background: none;
}
```

Dále je potřeba změnit pozadí, okraj a stínování u odkazů druhé úrovně.

```
.navbar-right .dropdown-menu {
  background: #333;
  border: none;
  box-shadow: none;
}
```

Podobně, jako ve vlastním řešení, je i zde funkce zobrazování a skrývání panelu v nejmenší variantě webu realizována pomocí JavaScriptu, který je však kompletně připraven v knihovnách samotného frameworku. Vložení tlačítka je znázorněno níže.

```
<button type="button" class="navbar-toggle collapsed"
data-toggle="collapse" data-target="#bs-example-navbar-collapse-1"
aria-expanded="false"></button>
```

Dalším problémem, se kterým se kodér může při tvorbě panelu setkat, je změna zlomového bodu, při kterém se panel překlopí do pouhého tlačítka. K tomu je potřeba vytvoření celého nového předpisu stylů po konkrétní mediální dotaz. Pokud by například

bylo nutné zalomit hlavní menu již při 1200px, dodatečný zdrojový kód by vypadal následovně:

```
@media (max-width: 1200px) {
  .navbar-header { float: none; }
  .navbar-toggle { display: block; }
  .navbar-collapse.collapse { display: none!important; }
  .navbar-nav { float: none!important; }
  .navbar-nav>li { float: none; }
  .navbar-collapse.collapse.in { display: block!important; }
  .collapsing { overflow: hidden!important; }
}
```

To je obvykle potřeba při obsáhlejších navigačních panelech, které obsahují větší množství odkazů, jež se nevejdou s brandem webu na jeden řádek.

4.5.3 Navigační panel pomocí Foundation

Velmi podobné řešení nabízí i Foundation v podobě komponenty *Responsive navigation*. Jeho výchozí aplikace na menu v návrhu vypadá následovně. Z toho je patrné, že je i zde nutné doplnit řadu kaskádových stylů pro změnu jeho vzhledu:

Obrázek 29 - Navigační panel ve Foundationu bez dodatečných stylů

Barva pozadí celého panelu při 100% šířce zajišťuje element se třídou `.mainmenu`.

```
.mainmenu {
  background-color: #333333;
}
```

Ve druhé řadě je nutné změnit hodnotu vlastnosti pozadí u třídy `.top-bar` a elementu `.top-bar ul`. Ty jsou v základu nastaveny na barvu *#e6e6e6*. Tento element zároveň bude udávat výšku celého panelu. Změna pozadí a výšky vypadá následovně:

```
.top-bar, .top-bar ul {
  background: none;
  min-height: 95px;
}
```

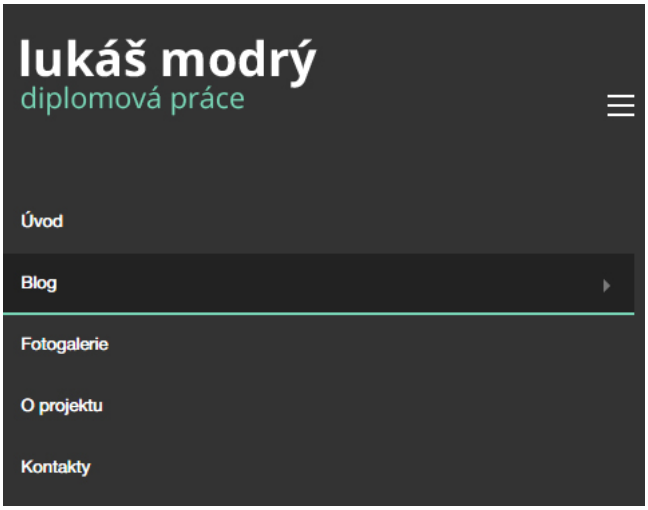
Pozadí je dále nutné nastavit i ve druhé úrovni menu. Zde je však nutné nastavit konkrétní barvu, protože přesahuje svůj obalový element.

```
.dropdown.menu .submenu {
  background: #333;
}
```

Tím je základní vzhled navigačního panelu vyřešen. Tento panel se však chová poněkud jinak při nejmenším zobrazení displeje. Oproti tomu, jak je to u ostatních dvou řešení, ve frameworku Foundation 6 nedochází k zarovnání jednotlivých odkazů pod sebe.

Obrázek 30 - Mobilní zobrazení navigačního panelu ve frameworku Foundation 6

Možné řešení tohoto problému je využívání starší verze frameworku, konkrétně verzi 5. Komponenta se jmenuje *Topbar* a je založena na podobně pojmenovaných třídách jednotlivých elementů. Při změně několika rozměrových parametrů vypadá následovně.



Obrázek 31 - Mobilní zobrazení navigačního panelu ve frameworku Foundation 5

Moment překlopení panelu do responsivní verze lze definovat pomocí hodnoty u atributu *data-hide-for* u elementu s třídou `.title-bar`. Ten předává hodnotu JavaScriptové funkci, zajišťující výměnu jednotlivých ovládacích prvků. V ukázkovém příkladu je zvolena hodnota *large*.

```
<div class="title-bar" data-responsive-toggle="hlavnimenu"
data-hide-for="large">
```

4.5.4 Doporučení k volbě navigačního panelu

Při volbě způsobu realizace hlavního navigačního panelu se vývojáři musejí zamyslet nad tím, jak složitý po stránce designu samotný panel bude. Ve většině případu není problém uchýlit se k řešení, které je k dispozici ve frameworku Bootstrap. Autor webu je však omezen způsobem responsivního zobrazení navigačního panelu při malém rozlišení.

Při potřebě vytvoření například panelu, který se bude vysouvat od levého okraje, je nutné změnit chování JavaScriptového kódu a kaskádových stylů, které tento efekt zajišťují. To může především pro méně zkušené kodéry být značně problematické. V takovém případě je vhodnější definovat celý navigační panel vlastními kaskádovými styly a JavaScriptem bez využití frameworku. Dalším obvyklým problémem při realizaci navigačního panelu pomocí frameworku je častá potřeba změnit moment překlopení, který nelze definovat jednoduchým nastavením, ale je nutné vytvořit další mediální dotaz, jenž tuto změnu zajistí.

Naopak navigační panel ve Foundationu je sice poměrně snadný na změnu vzhledu, jeho chování v mobilní verzi v poslední, v době psaní této práce, dostupné aktualizaci frameworku je však zcela nevhodné a v praxi by bylo nutné jeho chování zcela předefinovat pomocí kaskádových stylů. V takovém případě však využívání frameworku postrádá smysl. Řešením je zmíněné využití komponenty z předchozí verze 5. Je však značně komplikované využívat dvě verze jednoho frameworku, protože se mohou navzájem narušovat. Z toho důvodu je v ukázkovém přikladu využita poslední dostupná varianta[5].

Jednotlivá řešení vedle mediálních dotazů nevyužívají žádné jiné prvky, které by měly problémy s podporou v prohlížečích. Není tedy problém s jejich využíváním napříč zařízeními.

[5] Poslední verze frameworku ke dni 20. 1. 2016 – Foundation 6

4.6 Image slider

Přestože obrázkové měnící se hlavičky začínají být pomalu přežitkem, stále je mnoho klientů vyžaduje. Vytvoření funkčního slideru je většinou závislé na použití JavaScriptu. Ten však nemá podstatný vliv na jeho zobrazení v rámci responsivního designu, nebude tedy popisován.

4.6.1 Vlastní řešení pomocí pluginu

Při realizaci obrázkového slideru bez použití frameworku byl využit jednoduchý JavaScriptový plugin s názvem *Responsive Slides*. (Salminen, 2015) Samotný prvek je vytvořen pomocí obyčejného netříděného seznamu:

```
<ul class="rslides rslides2">
  <li>
    <img src="img/image-1.jpg" alt="Image 1">
    <div class="caption">…</div>
  </li>
  <li>
    <img src="img/image-2.jpg" alt="Image 2">
    <div class="caption">…</div>
  </li>
</ul>
```

Základní třídou je `.rslides`, která určuje šířku celého slideru, v ukázkovém případě tedy 100%. Pro responsivní zobrazení slideru jsou dále klíčové styly nastavené k jednotlivým obrázkům:

```
.rslides img {
  display: block;
  height: auto;
  float: left;
  width: 100%;
}
```

Ty zajišťují, že se obrázek bude proporcionálně přizpůsobovat obalovému elementu. Je však nutné dbát na to, aby byl obrázek dostatečně velký, jinak dojde k jeho rozostření při příliš vysokém rozlišení obrazovky. Ostatní vlastnosti kaskádových stylů řeší skript. Ten především mění nastavení stylů jednotlivých položek seznamu, které následně fungují jako samostatné slidy.

K výchozímu slideru je následně ještě dodělán element *div* se třídou `.caption`, který slouží k umístění textu do slideru.

```
.rslides .caption {
  position: absolute;
  top: 20%;
  width: 100%;
}
```

Možností pro vytvoření responsivního slideru je však nepřeberné množství. Pluginů, realizujících tento prvek je k dispozici velmi mnoho a vývojářům tedy stačí si vybrat ten, který jim bude nejvíce vyhovovat a který bude nejvíce odpovídat konkrétním požadavkům daného projektu.

4.6.2 Carousel v Bootstrapu

V Bootstrapu je vývojářům připravena komponenta s názvem Carousel. Ta je vytvořena pomocí elementů *div* s třídou `.item`, které jsou vyměňovány pomocí JavaScriptu. Součástí komponenty je i prostor po responsivní popisek se třídou `.carousel-caption`. V případě delšího textu je však s tímto popiskem problém při malém rozlišení obrazovky.

Obrázek 32 - Image slider v Bootstrapu

Samotná přizpůsobitelnost obrázků v jezdci je realizována pomocí těchto vlastností:

```
.carousel-inner .item a img, .carousel-inner .item img {
  display: block;
  max-width: 100%;
  height: auto;
}
```

Z toho je patrné, že obrázek vždy vykryje celou šířku obalového elementu a jeho výšku přizpůsobí podle proporcí, které jsou dopočteny podle vygenerované šířky. Takto vytvořený image slider se bude přizpůsobovat svému obsahu a při použití různě velkých obrázků tak nebude jednotný. Je tedy vhodné používat stejně velké obrázky, které budou do hlavičky webu vkládány. Další možností je přestylování elementu tak, aby měl jasně definované rozměry a veškerý přetékající obsah skryl. To by však byla značně komplikovaná práce.

4.6.3 Orbit ve Foundationu

Podobná komponenta je obsažena i ve frameworku Foundation. Vytvořena je podobně jako ve vlastní variantě, tedy jako klasický seznam s ovládacími prvky, popiskem a obrázkem.

Obrázek 33 - Image slider ve Foundationu

Umístění obrázku je realizováno stejně, jako v Bootstrapu. Obrázek se tedy přizpůsobuje 100% šířce obalového elementu a výška je proporcionálně dopočítána. Výměna jednotlivých slidů je opět realizována pomocí JavaScriptu.

Tento slider však obsahuje chybu zobrazení výšky prvního obrázku. Ta může nastat například při změně velikosti okna, či při otočení mobilního zařízení z polohy *portrait* do polohy *landscape*.

4.6.4 Doporučení při volbě image slideru

Při realizaci webové prezentace, navržené přímo pro využívání některého frameworku je usnadnění využívat příslušnou komponentu. Vývojáři jsou však často omezeni právě efektem prolínání, velikostí samotného slideru, případně samotnou funkcí.

Jakékoliv přizpůsobování těchto komponent je poněkud náročné a nepraktické. Z toho důvodu je ve většině případů vhodné místo jejich využívání zvolit alternativu v podobě některého z dostupných pluginů vytvořených za pomoci knihovny JQuery. Díky tomu si mohou vývojáři ušetřit značné množství práce, zapříčiněné přizpůsobováním komponent frameworků. Naopak výhodou při používání těchto komponent je ověřená funkčnost. Z obou testovaných frameworků vychází lépe alternativa z Boostrapu, která nevykazuje v základním nastavení žádné chyby zobrazení. Jeho konkurent v podobě Orbitu ve Foundationu vykazuje chyby v zobrazení při změně velikosti okna, je tedy nutné ho manuálně opravit, případně nahradit jiným typem slideru. Tím je však potlačena podstata frameworku.

4.7 Responsivní obrázky

Velmi diskutovaným odvětvím v rámci responsivního designu je práce s obrázky a jinou grafikou. Vývojáři se při tvorbě webových stránek musí potýkat s několika základními problémy. Mezi ty patří například vhodná velikost zobrazovaných obrázků, jejich datová náročnost, případně jejich hodnota počtu pixelů na palec. Technologie pro tvorbu webových stránek obsahují několik možností, jak publikovat obrazové soubory na internetu.

4.7.1 Relativní velikost obrázku

Nejzákladnější způsob, kterým lze umístit obrázky na web a zajistit jejich responsivní zobrazení, je nastavení velikosti obrázku pomocí relativních jednotek, tedy v procentech. Tato metoda zpravidla pracuje pouze s jednou variantou obrázku, který následně přizpůsobuje velikosti obalového elementu. V ukázkovém příkladu je požadavkem při velkém rozlišení umístit obrázek na poloviční šířku obsahu a zbytek prostoru nechat obtékat textem. Při malém rozlišení dále vymezit celou šířku obsahu pro tento obrázek.

K umístění takového obrázku stačí použití klasického elementu *img*, který vypadá následovně a je pouze obohacen o třídu, jež tento obrázek jednoznačně identifikuje.

```
<img src="img/1200.jpg" alt="Popisek obrázku" class="relativni">
```

Nastavení této třídy v kaskádových stylech následně umožňuje, aby se obrázek automaticky přizpůsoboval obsahu. Obrázek bude v základním zobrazení, což je v rámci

metody *Mobile First* nejmenší definované rozlišení, pokrývat celou šířku svého obalového elementu. Samotné styly jsou pak napsány takto:

```
.relativni {
  width: 100%;
  max-width: 1200px;
}

@media (min-width: 768px) {
  .relativni {
    width: 50%;
    float: left;
  }
}
```

Obrázek 34 - Responsivní obrázek v článku

Při používání relativních jednotek se vývojáři musejí potýkat ještě s jedním problémem. Tím je maximální velikost zobrazovaného obrázku. Fotografie s názvem *1200.jpg*, která je použita ve výše zmíněném příkladu, má rozměry 1200 pixelů na šířku a 800 pixelů na výšku. Z toho vyplývá, že kdyby tento obrázek nebyl omezen vlastností *max-width* právě na rozměr 1200px, mohla by nastat situace, že by tento obrázek v rámci procentuálního vykrývání obalového elementu tuto velikost překročil a začal by ztrácet na své obrazové kvalitě. Tato situace však může nastat i v opačném případě, kdy při příliš velkém zmenšení obrázek začne ztrácet své detaily. Tento problém lze vyřešit atributem

min-width, který však může způsobit problémy v momentě, kdy se obalový element stane menším jak tato limitní hranice velikosti obrázku. V takovém případě by obrázek začal přetékat svůj obalový element a způsobil by jak rozhození samotného vzhledu webové stránky, tak její celkové responsivní zobrazení. V tomto konkrétním příkladu by způsobil zobrazení horizontálního posuvníku, který je nežádoucí.

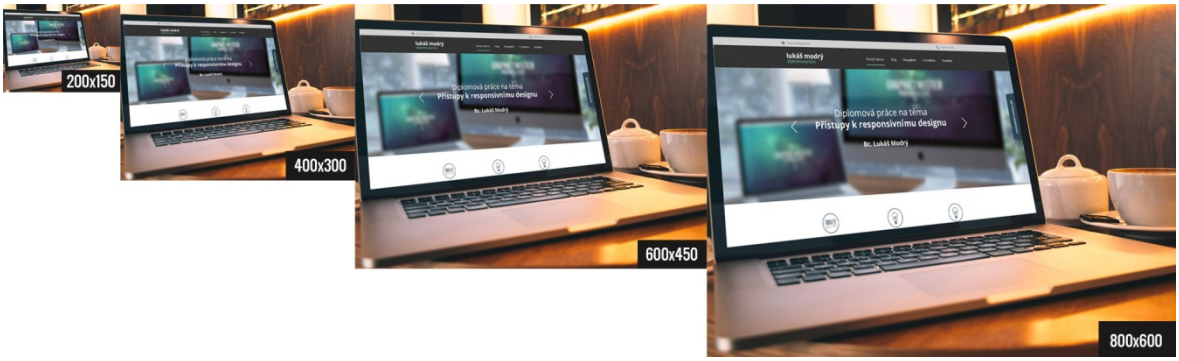
Velkou výhodou je úplná podpora ve všech webových prohlížečích, které podporují kaskádové styly. Naopak nevýhodou při tomto způsobu zobrazování přizpůsobitelného obrázku je jeho datová náročnost. Pro pokrytí co největší škály možných zobrazení je nutné použít co největší možný obrázek. V případě Full HD fotografií a větších se pak jedná o obrazová data, která mohou svou velikostí přesáhnout i 1 MB, což je nepřípustné pro zařízení, která jsou k internetu připojena prostřednictvím 2G mobilní sítě a omezena datovou kapacitou FUP.

4.7.2 Srcset

Právě problém, kdy nestačí jeden obrázek pro všechna rozlišení, řeší nová vlastnost jazyka HTML5 s názvem *srcset* a k ní přidružená vlastnost *sizes*. Tyto vlastnosti jsou prioritně určeny jako další rozšiřující parametry pro element *img*.

První jmenovaná vlastnost umožňuje definovat více zdrojů k zobrazení obrázku v závislosti na rozlišení průhledu. V ukázkovém přikladu jsou k dispozici 4 různé obrázky pro jednotlivá rozlišení:

- *200x150.jpg* pro velikost průhledu 250px
- *400x300.jpg* pro velikost průhledu 400px
- *600x450.jpg* pro velikost průhledu 600px
- *800x600.jpg* pro velikost průhledu 800px



Obrázek 35 - Varianty obrázků pro vlastnost Srcset

Element *img* je dále rozšířen ještě o vlastnost *sizes*, která určuje velikost zobrazení konkrétního obrázku v závislosti na vyhodnocení mediálního dotazu. To však pracuje s velikostí celého průhledu, nikoli s velikostí obalového elementu, což je značně nepraktické. Především pak v okamžiku, kdy je požadavkem omezit obrázek na 50% obalového elementu, u kterého však není známo, kolik procent průhledu v daném momentě zabírá. Z toho důvodu je nutné element *img* dále omezovat pomocí kaskádových stylů a zlomových bodů - obdobně jako v předchozím příkladu.

Celý element *img* pak vypadá následovně:

```
<img src="img/800x600.jpg"
     srcset="img/200x150.jpg 250w,
             img/400x300.jpg 400w,
             img/600x450.jpg 600w,
             img/800x600.jpg 800w"
     sizes="(min-width: 768px) 50%, 100%"
     alt="Popisek obrázku">
```

Z ukázky je patrné, že samotný prvek je dále obohacen ještě o standardní vlastnosti, kterými jsou *src*, pro zobrazení výchozího obrázku při nepodpoře nových vlastností prohlížečem, případně pro zobrazení obrázku při nesplnění žádné z podmínek, a dále atribut *alt*, který definuje popisek obrázku při jeho nenačtení.

Výhodou je, že v rámci vyhodnocení podmínky je stažen pouze jeden vhodný obrázek, který bude zobrazen, což v praxi šetří data, potřebná k zobrazení samotného obrázku. Naopak nevýhodou je, že prohlížeč se snaží stále zobrazovat největší variantu obrázku, což však může být zapříčiněno nedostatečnou podporou v prohlížečích.

Nové vlastnosti mají zásadní problém s podporou v prohlížečích. Největším nedostatkem je nulová podpora v prohlížeči Internet Explorer. Dalšími nepodporujícími aplikacemi pro prohlížení webu jsou starší verze Android Browseru a také mezi uživateli smartphonů oblíbená Opera Mini. Řešením je využití skriptu, který by funkci těchto vlastností doplnil. Tím se však web stává závislý právě na používání JavaScriptu, který mají někteří uživatelé vypnutý. V příkladu byl využit *polyfill* s názvem *PictureFill*. (Jehl, 2015) Ten doplňuje právě požadovanou funkčnost. Jeho nevýhodou však je, že v rámci doporučení není vhodné používat alternativní atribut *src* pro načtení výchozího obrázku

z důvodu vícenásobného stahování datových souborů. To však může vést k potížím se zaindexováváním obrázku prostřednictví Googlu.

4.7.3 Picture

Alternativou k předchozí variantě s používáním nových vlastností je zcela nový element HTML5 s názvem *picture*. Ten se skládá ze tří základních prvků:

- `<picture>` - jakožto obalového elementu celého obrázku
- `<source>` - určeného k definování zdrojových obrázků s mediálním dotazem a cesty k danému souboru
- `<img>` - pro definování alternativního výchozího obrázku

Samotný prvek funguje na základě vyhodnocení mediálního dotazu. Na jeho základě zobrazí požadovaný obrázek. I zde je využívána vlastnost *srcset*. Samotný zápis elementu v ukázkovém příkladu vypadá následovně:

```
<picture alt="popisek obrázku">
  <source media="(max-width: 767px)"
      srcset="img/400x300.jpg">
  <source media="(min-width: 768px) and (max-width: 991px)"
      srcset="img/600x450.jpg">
  <source media="(min-width: 992px) and (max-width: 1169px)"
      srcset="img/800x600.jpg">
  <img src="img/1200.jpg" alt="popisek obrázku">
</picture>
```

Z příkladu je očividné, že při rozlišení obrazovky menší jak 768px bude zobrazen nejmenší možný obrázek, dále větší pro rozlišení od 768 do 991px a největší od 1170px šířky průhledu vždy v závislosti na splnění konkrétního požadavku. Oproti své alternativě tento element funguje podstatně spolehlivěji a jeho definice je sice delší, avšak z praktického hlediska jednodušší a přehlednější. Dále disponuje širší škálou možností zobrazení na základě různých mediálních dotazů. Obdobně, jako v předchozím příkladu, je i zde stažen pouze jeden momentálně požadovaný obrázek. V případě změny velikosti okna jsou dále stahovány ostatní velikosti pro různá zobrazení.

I zde se však vývojáři musí potýkat s nedostatečnou podporou v prohlížečích a jejich nativní podpora je otázkou několika měsíců až let. V době psaní této práce element není podporován v prohlížečích Internet Explorer, Safari a značném množství mobilních

prohlížečů. Samotná funkčnost je v těchto nepodporujících aplikacích opět zajištěna prostřednictvím polyfillu *Picturefill*, stejně jako v předchozí ukázce.

Pro využívání obrázků s různou velikostí DPR (které je popsáno v následující podkapitole) lze pro element *picture* vytvořit speciální mediální dotaz, na jehož základě lze zhodnotit právě tuto hodnotu:

```
media="(min-device-pixel-ratio: 2), (min-resolution: 192dpi)"
```

4.7.4 Device Pixel Ratio

Velikost samotných obrázků však není jediný problém, s nímž se musí vývojáři webových stránek potýkat. Se stále rostoucím rozlišením obrazovek při snižování velikosti jejich úhlopříčky vzniká zásadní problém s kvalitou zobrazovaných obrázků. Mnoho dnešních mobilních telefonů a tabletů disponuje obrazovkami s rozlišením větší strany přesahující velikost 1200 pixelů. V praxi to pak znamená, že například při rozlišení tabletu 1920 na 1200 pixelů při 10,1“ úhlopříčce se texty a jiné prvky na webu stávají nečitelnými. Je to z důvodu jejich optické velikosti, která je poskytnuta uživateli. Na příkladu z tlačítka, které je na normálním 24“ monitoru velké 3 cm je na tomto tabletu tlačítko velké pouze 1,25 cm. Přestože na digitální data není standardem používat klasické měrné jednotky, je nutné si uvědomit poměr velikosti reálných pixelů na palec a to, jak velký výstup ve finále uživatel ve skutečnosti dostane.

Mnoho moderních zařízení s přístupem k internetu využívá jednotky *device pixel ratio*, tedy poměr skutečných a virtuálních pixelů. Ty ve skutečnosti říkají, o kolik procent je samotný obsah na obrazovce zvětšen uživateli vůči reálné velikosti v pixelech. Zmíněný tablet má nativně nastavenou hodnotu *DPR* rovnu 1,5, tedy 150% zvětšení veškerého obsahu, který je jeho vlastníkovi zobrazen.

Při publikování obrázků na internetu je tedy nutné, aby se vývojáři zamysleli nad tím, jak moc bude daný obrázek na jednotlivých zařízeních vůči své skutečné velikosti zvětšen a na základě toho zamezili ztrátě obrazové kvality.

K řešení tohoto problému slouží rozšíření vlastnosti *srcset* o parametr, který poskytuje možnost vyhodnocení poměru pixelů. Zápis vypadá následovně:

```
<img srcset="img/text72dpi.jpg,
             img/text150dpi.jpg 1.5x,
             img/text300dpi.jpg 2x">
```

Pro klasické zobrazení, kde se počet reálných pixelů rovná počtu virtuálních pixelů, bude použit obrázek s hodnotou DPI (*Dot per Inch*) rovnou jedné. Pro DPR=1,5 bude zobrazen podrobnější obrázek s DPI=150 a pro 200% přiblížení pak obrázek s DPI=300. Je však nutné si uvědomit datovou náročnost jednotlivých obrázků. Je rovněž možné přidat další podmínky pro další zařízení, která disponují 4K displejem.

V praxi by pak obrázek se 72dpi na obrazovce s hodnotou DPR=2 oproti své původní ostrosti vypadal asi takto:

Obrázek 36 - Různá DPR při použití stejného obrázku

Při používání hodnot DPR jsou však vývojáři opět omezeni podporou v jednotlivých prohlížečích. Hodnota je v době psaní této práce podporována pouze v poslední verzi prohlížeče Google Chrome a u poslední verze prohlížeče Android Browser. Možností je vyhodnocení samotné hodnoty pomocí JavaScriptu a následné generování vhodného obrázku pomocí výstupní hodnoty.

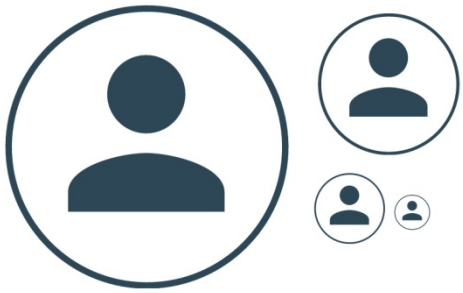
4.7.5 SVG

Ne vždy však tvůrcům webových stránek stačí různé alternativy velikostí jednotlivých obrázků. Především v momentě využívání různých infografik, sad ikon a jiných takovýchto designových prvků, je nutné se uchýlit k práci s vektorovými obrázky.

Formát SVG lze vyvářet v různých vektorových aplikacích nebo přímo prostřednictvím zdrojového kódu. V ukázkovém příkladu je použita první zmíněná varianta. Samotné vektorové soubory lze vkládat prostřednictvím klasických principů pro vložení jakéhokoliv jiného typu obrázku.

```
<img src="css/icon-autor.svg" alt="Autor">
```

Jejich nastavení v rámci kaskádových stylů definuje jejich rozměr v relativních jednotkách. Díky vektorové grafice však samotné obrázky zůstávají vždy dokonale ostré a detailní. Příklad použití souborů SVG je v ukázkovém webu na pozici ikon u hlavních rozcestníkového panelu.



Obrázek 37 - použití formátu SVG

Další velkou výhodou tohoto formátu je možná editovatelnost obrazových dat pomocí zdrojového kódu. Obrázky jsou tak strojově čitelné a lze je bez problému indexovat pomocí robotů vyhledávačů.

Vektorové obrázky ve formátu *SVG* nejsou podporovány pouze v prohlížeči Internet Explorer 8 a starší.

4.7.6 Shrnutí práce s obrázky

Jednotlivé přístupy k zobrazení responsivních obrázků jsou na první pohled velmi podobné, je však nutné se zamyslet nad tím, jak budou uživatelé stránku zobrazovat.

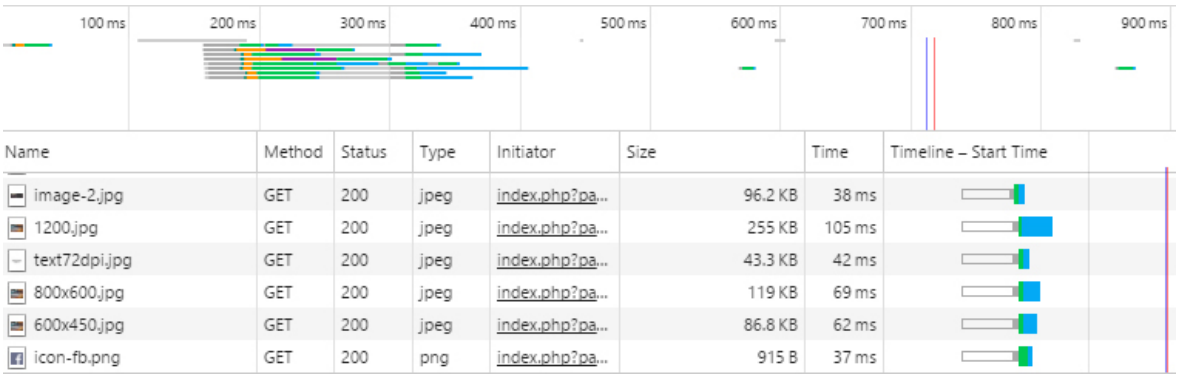
Ve všech ohledech je v rámci podpory v prohlížečích nejvhodnější zastaralá varianta s relativními jednotkami. V případě, že vývojáři necílí na uživatele s pomalým připojením k internetu a použijí rozumně velké obrázky, je tato metoda nejefektivnější i nad rámec podpory, protože její implementování do webové stránky je jednoznačně nejjednodušší. Odpadají náklady jak na vytváření jednotlivých alternativ obrázků, přičemž zápis a kontrola nad samotným obrázkem spočívají v prostém definování zažitých kaskádových stylů. S nadsázkou se dá říci, že obrázky do velikosti 500 kB jsou vhodné pro používání pro veškeré typy připojení, přestože při opravdu slabém signálu bude s jejich stažením problém.

Naopak při nutnosti limitovat datovou velikost obrázků například u webů, kde je zobrazení konkrétního obrázku opravdu nutností, jsou nové možnosti jazyka HTML užitečné. V rámci funkčnosti a náročnosti na implementaci je vhodnější využívání elementu *picture*, který přestože nativně není podporován v žádném prohlížeči, s dodatečným polyfillem funguje spolehlivěji jak jeho alternativa v podobě vlastností elementu *img*. Jedná se především o bezproblémové načítání obrazových dat v závislosti

na rozlišení obrazovky. Pro dosažení stejné funkcionality je však vhodné tento element doplnit sadou kaskádových stylů pro definování velikosti zobrazovaného obrázku.

V praxi tedy vývojář docílí kombinace nové technologie doplněné starou technikou pro zobrazování obrázků. Tím však získá úplnou kontrolu nad obsahem.

Samotné použití těchto přístupů je znázorněno ve statistice stažených souborů při rozlišení průhledu 750px na šířku.



Obrázek 38 – Statistika stažení responsivních obrázků

Ze statistiky je patrné, že základní přístup stáhl obrázek 1200.jpg, tak, jak bylo popsáno. Dále je však vidět rozdíl mezi obrázkem staženým elementem *img* s atributem *srcset* a jeho alternativou v podobě značky *picture*. Přestože, by oba přístupy měly vyhodnotit stejný obrázek, není tomu tak. První jmenovaná alternativa se snaží zobrazovat obrázek větší, než je definováno.

Pro načítání obrázků na jemnějších obrazovkách lze využívat obrázky na základě vyhodnocení hodnoty DPR. Využívání tohoto přístupu však postrádá větší smysl, pokud není požadavkem zobrazovat velmi přesné obrázky a text v obrázcích na velmi jemných obrazovkách. V rámci obyčejných uživatelů lze využívat normální obrázky bez nutnosti větší hustoty pixelů na palec, protože moderní displeje si s tím bez problémů poradí a obrázky vykreslí i tak dostatečně ostré. Tím se ušetří i datová náročnost.

K zobrazení ikon a jiného jednoduchého obrazového materiálu, který se dá snadno realizovat pomocí vektorové grafiky, je vždy vhodné tyto soubory vkládat do webové stránky za pomoci formátu *SVG*, který zajistí kvalitu zobrazení na všech zařízeních s libovolným rozlišením. Tyto soubory jsou dále snadněji upravitelné a lze s jejich pomocí snáze definovat obsah stránek například pro nevidomé uživatele.

4.8 Práce s formuláři

Responsivní formuláře jsou velmi užitečným nástrojem při vytváření kvalitních přizpůsobujících se stránek. V ukázkovém příkladu je vytvořen jednoduchý kontaktní formulář v podstránce *kontakty*, ve kterém jsou využita základní textová pole, využívaná na mnoha webových prezentacích.

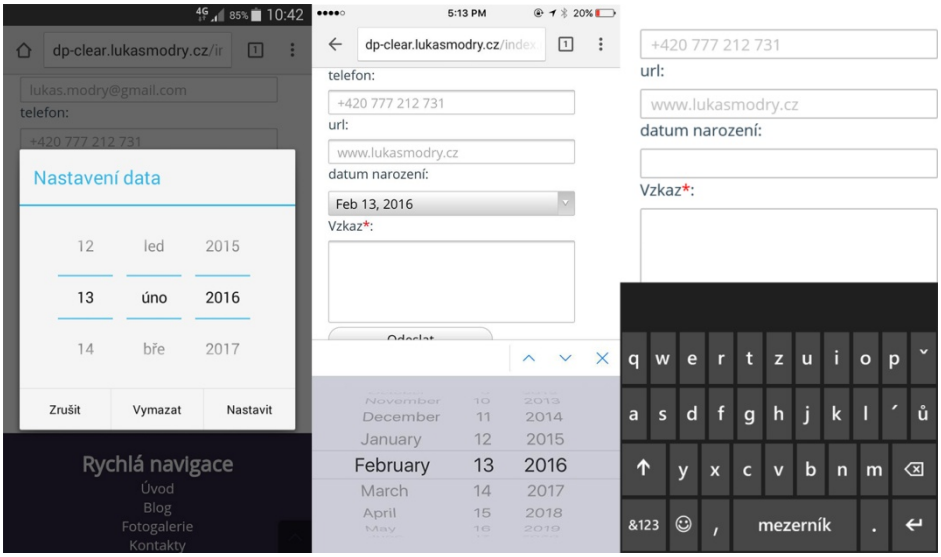
Jedná se o prvky *input* s hodnotou typu:

- *email* – pro vkládání e-mailové adresy
- *tel* – pro vkládání telefonního čísla
- *url* – pro vkládání webové adresy
- *date* – pro vkládání datumu

Výhodou těchto prvků jazyka HTML5 je poskytnutí speciální klávesnice na dotykových obrazovkách, což usnadňuje ovládání těchto formulářů. Velký vliv to má především u vyplňování obsáhlých formulářů, které se používají například na elektronických obchodech pro realizování objednávky. Klasické textové pole s plnohodnotnou klávesnicí může mnoho uživatelů odradit od samotné objednávky, protože vyplňování různých speciálních znaků a číslic je pro ně příliš zdlouhavé. To má za následek nižší procento konverzí, tedy i nižší příjmy společností, které webové stránky provozují.

```
<input type="email" name="email">
<input type="tel" name="telefon">
<input type="tel" name="url">
<input type="date" name="datum">
```

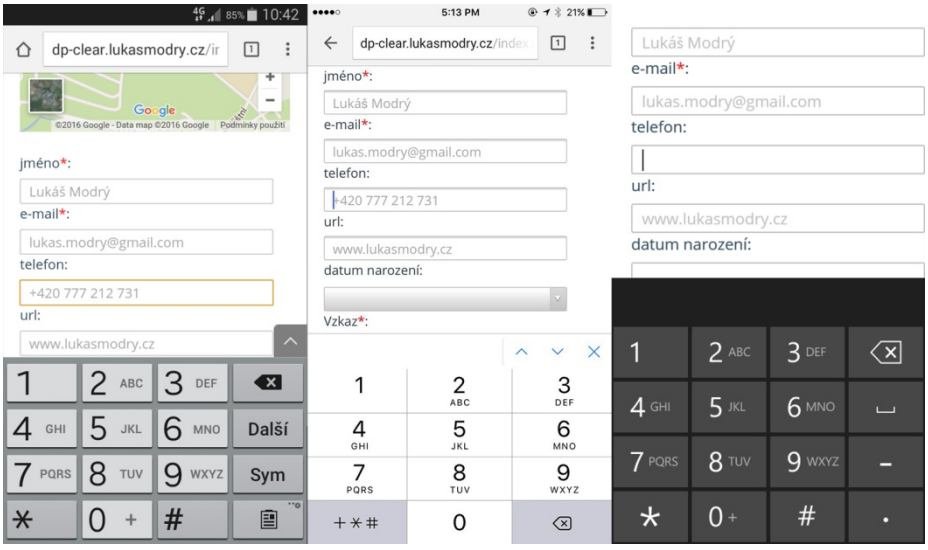
Na jednotlivých zařízeních je však nutné počítat s tím, že jednotlivé typy disponibilních klávesnic jsou značně odlišné a nemusí tak korespondovat s grafickým návrhem. Jako ukázka je použito odlišné zobrazení pole pro vkládání data, které je napříč zařízeními zcela odlišně, byť jeho funkcionalita je totožná.



Obrázek 39 - Input typu *date* na Android 4.4.4, OS X 10.8.5 a Windows 8.1

Z obrázku je patrné, že jednotlivé operační systémy využívají grafiku, která je pro daný systém výchozí. V prohlížeči pod Windows 8.1 textové pole typu *date* není podporováno a je tak načtena klasická klávesnice pro zadávání textu.

Podobnou ukázkou může být i pole pro vkládání telefonního čísla, kde jsou klasické plnohodnotné znakové klávesnice nahrazeny číselnou. Ty jsou na jednotlivých zařízeních velmi podobné a rozdíl je jen v jejich vzhledu, případně rozložení doplňujících kláves.



Obrázek 40 - Input typu *tel* na Android 4.4.4, OS X 10.8.5 a Windows 8.1

Jejich využívání je zcela bezproblémové. V okamžiku, kdy nejsou podporovány, zařízení a prohlížeče automaticky zobrazí výchozí klávesnici jako u obyčejného textového

pole. Samotné prvky jsou však u většiny zařízení již v době psaní této práce podporovány. Je tedy vhodné je používat.

4.9 Práce s tabulkami

Jedním z nejkomplikovanějších prvků jazyka HTML pro responsivní design jsou bezesporu tabulky. Zásadní problém spočívá v nemožnosti zalamování jednotlivých buněk pod sebe, ať už kvůli ztrátě kontextu, tak kvůli technologické stránce samotného elementu *table*. Vývojáři jsou v praxi omezeni jen dvěma možnostmi. Tedy relativní šířkou jednotlivých sloupců, případně možností umožnit tabulce přetékání obsahu.

První zmíněná možnost je praktická do momentu, než se samotné rozlišení pracovní plochy prohlížeče zmenší na příliš malý rozměr.

```
<table class="relativni-sloupce">
  <thead>…</thead>
  <tbody>
    <tr>
      <td>…</td>
      <td>…</td>
      …
    </tr>
    …
  </tbody>
</table>
```

A k ní přiřazené kaskádové styly:

```
.relativni-sloupce { width: 100%; }
.relativni-sloupce tr td { width: auto; }
```

Takto zapsaná tabulka se bude vhodně zobrazovat pouze na velkých obrazovkách s vysokým rozlišením. Na malém rozlišení se však její obsah nevejde do svého obalového elementu a tabulka začne přetékat, což způsobí chybu v zobrazení celého webu a vytvoří horizontální posuvný panel.

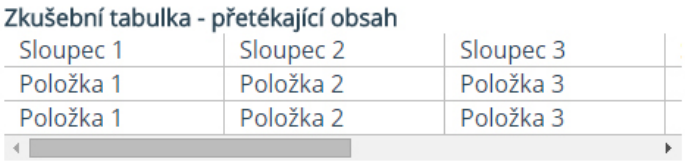
Obrázek 41 - Tabulka pomocí relativních rozměrů

Tento problém může vyřešit povolení přetékání obsahu s přiřazeným horizontálním scroll barem.

```
<table class="overflow-sloupce">
  …
</table>
```

Použitá třída má následně nastavenou šířku pro pokrytí celého možného obsahu a dále požadavek na vytvoření horizontálního panelu, pro jehož funkci je nutné nastavit dále vlastnost *display* na *block*. Panel se zobrazí v momentě, kdy se samotná tabulka nevejde do svého obalu. Kaskádové styly jsou následující:

```
.overflow-sloupce-obal {
  width: 100%;
  overflow-x: auto;
  display: block;
}
```



Obrázek 42 - Responsivní tabulka s přetékajícím obsahem

Podobnou metodu využívají i oba porovnávané frameworky. V Bootstrapu je třída pro přetékání definována bez vlastnosti *display*, je tedy nutné ji aplikovat na obalový element celé tabulky. S tím je spojena nutnost vytvoření dalšího elementu.

```
<div class="table-responsive">
  <table class="table">
    …
  </table>
</div>
```

Framework Foundation přetékající tabulky řeší zcela stejně, jako je v ukázkovém příkladu výše a to pomocí třídy *scroll*.

Tato metoda je vesměs jediná použitelná pro vytváření responsivních tabulek. Při používání frameworků je jednodušší použití u druhého jmenovaného, protože zde není nutnost využívat dvě různé třídy pro dosažení tíženého efektu.

Nastavení vlastnosti *overflow* na hodnotu *auto* je podporované ve všech prohlížečích a není tedy žádný problém a omezení s jejím využíváním. Pro použití této metody je ale vhodné respektovat určení použití tabulky, tedy element pro zobrazení hodnot do sloupců a řádků. Je zcela nevhodné tabulku používat pro strukturování samotného obsahu a vzhledu stránky, jako tomu bylo před několika lety u takzvaných tabulkových layoutů.

4.10 Zobrazování a skrývání prvků

Jednou ze spíše krajních možností pro zobrazování responsivních elementů je princip zobrazování a skrývání jednotlivých prvků. K tomu slouží zcela základní vlastnosti kaskádových stylů, která byla použita již mnohokrát, tedy *display*, tentokrát s hodnotou *none*.

Tato metoda na základě rozlišení obrazovky některé prvky skryje a jiné zobrazí. Ve zdrojovém kódu stránky se však načtou a je tedy zcela nevhodné tuto metodu využívat například na zobrazování a skrývání různě velkých obrázků.

V praktickém příkladu je ukázka v horní šedé liště s informacemi o aktuálně aktivním mediálním dotaze, kde je struktura složena z elementů *span* s přiřazenými třídami pro jednotlivé podmínky.

```
<span class="default">Výchozí styly</span>
<span class="small">Minimální rozměr průhledu: 768px</span>
<span class="medium">Minimální rozměr průhledu: 992px</span>
<span class="large">Minimální rozměr průhledu: 1170px</span>
```

Tyto 4 třídy jsou v kaskádových stylech nastaveny velmi jednoduše. Ve výchozím zobrazení je zobrazen pouze základní element s třídou *default* a ostatní jsou skryté.

```
.default {
  display: block;
}

.small, .medium, .large {
  display: none;
}
```

Následně se pak tyto hodnoty mění napříč mediálními dotazy. Z názvů tříd je patrná jejich posloupnost zobrazování, tedy zobrazení třídy *small* a skrytí *default*, dále pak zobrazení třídy *medium* a skrytí *small*, a tak dále.

Obrovskou nevýhodou je to, že se všechny 4 elementy načtou i v momentě, že jsou skryté. V případě používání této metody na datově obsáhlejším obsahu, jako jsou obrázky, dlouhé texty, videa, a podobně, značně naroste datová náročnost celé stránky, což je nevhodné pro používání na přenosných zařízeních s mobilním internetem.

Obdobné třídy jsou k dispozici v obou frameworcích a fungují na zcela stejném principu. Jediný rozdíl je ve zvoleném zlomovém bodě. V Bootstrapu byly využity třídy *visible-xs*, *visible-sm*, *visible-md* a *visible-lg*. Ve Foundationu se jedná o *show-for-small-only*, *show-for-medium-only*, *show-for-large-only* a *show-for-xlarge-only*.

5 Výsledky a zhodnocení

Nároky dnešních uživatelů internetu na webové prezentace jsou obrovské. Většina návštěvníků stránek využívá k přístupu k webu větší množství různých zařízení, ať už jsou to mobilní telefonu, tablety, notebooky, nebo stolní počítače. S rostoucími nároky uživatelů se web neustále vyvíjí. Je to velmi nestálé prostředí s rychle se měnícími požadavky a trendy. Pro vývojáře je pak velmi náročné držet krok s dobou a přizpůsobovat se možnostem moderních technologií.

5.1 Porovnání jednotlivých přístupů

Z praktického hlediska se dá tvrdit, že využívání jednotlivých prvků je vždy závislé na daném projektu. Nelze tedy jednoznačně určit, zdali je lepší ta, či ona komponenta z konkrétního frameworku, či je lepší zvolit vlastní přístup. Vždy je potřeba se zamyslet nad tím, jak má finální webová stránka vypadat a fungovat, jací uživatelé ji budou navštěvovat a v jakých situacích. V celkovém shrnutí jednotlivých testovaných částí webové stránky byly zjištěny výsledky, které jsou zaneseny do následující tabulky a následně rozepsaný.

	Bez frameworku	**Bootstrap 3.6.6**	**Foundation 6**
Viewport	Závislé na konkrétním projektu		
Struktura webu	Čistý zdrojový kód, Složitá realizace	Nečistý zdrojový kód, Velmi jednoduchá realizace	Nečistý zdrojový kód, Jednoduchá realizace
Mediální dotazy	Libovolné	*screen, all*	*screen, all*
Navigační panel	Libovolný, dle projektu	Omezený, Složitý na redesign	Omezený, Složitý na redesign
Image Slider	Libovolný, dle projektu	Velmi omezený vzhled	Omezená možnost změny efektů
Responsivní obrázky	Relativní rozměry, <*picture*> se chová lépe	Relativní rozměry	Relativní rozměry
Formuláře	Vždy využívat nové formulářové typy v HTML5		
Tabulky	Přetékání obsahu (horizontální slider)		
Zobrazování prvků	Na základě mediálních dotazů vlastností *display*		

Tabulka 1 - Porovnání jednotlivých prvků

Definování viewportu poskytuje značné množství různých výchozích zobrazení webové stránky a nastavení celého průhledu samotného prohlížeče. V praxi je však všeobecně dostačující využívání jen pár základních parametrů, které poskytnou vhodné zobrazení na většině dostupných zařízení.

Při vývoji responsivních webových stránek je nejdůležitějším krokem prvotní návrh samotné struktury celého webu a rozmístění jednotlivých prvků na stránce. Z praktického hlediska je velmi složité určit, který z přístupů je tím nejvhodnějším, protože každý z testovaných je vhodný pro jiný typ projektů. Obecně však lze říci, že pro čistý a přehledný zdrojový kód stránek je nejvhodnější alternativou vlastní přístup bez použití předdefinovaných frameworků. Je však nutné dbát na sémantické pojmenování jednotlivých tříd kaskádových stylů, či samotných HTML elementů, ze kterých je webová prezentace poskládána. To je především důležité v momentě, kdy se strukturou stránek budou v budoucnu pracovat další osoby.

Tento problém odpadá při využívání frameworků. Především proto, že zdrojové kódy těchto předdefinovaných stylů jsou všeobecně známé a většina vývojářů se s nimi setkává u mnoha projektů. S využíváním frameworků však narůstá potřeba přestylovávání již hotových kaskádových stylů. Při definování struktury je však méně náročné změnit několik vlastností CSS, než psát vlastní styly pro celý web, což je podstatně levnější alternativa.

Zdánlivě nejvhodnějším přístupem pro rozvržení obsahu webové stránky je kombinace vlastních stylů, doplněných o možnosti, které poskytují frontendové frameworky. V takovém případě je však nutné zvolit vhodný framework pro daný projekt a ten následně využívat i pro ostatní komponenty webové stránky. Při používání více knihoven stylů narůstá riziko kolizních tříd, které se budou vzájemně narušovat, a následně bude podstatně složitější definovat samotný vzhled celé prezentace.

Z dvou testovaných frameworků vychází lépe Bootstrap od společnosti Twitter, který pracuje především s absolutními zlomovými body, což umožňuje výrazně lepší kontrolu nad obsahem. Rozvržení stránky však není jediným responsivním aspektem při vývoji stránek. Vývojáři se dále musí potýkat s množstvím různých komponent, které jsou součástí většiny dnes používaných webových stránek. Z valné většiny se tyto prvky snaží

řešit jmenované frontendové frameworky. Při jejich využívání však vzniká značné množství omezení jak vzhledu, tak funkcionality jednotlivých komponent.

S rozložením webové stránky a jejím vzhledem je úzce spjato využívání mediálních dotazů na zařízení a jeho parametry. V praxi se většina vývojářů spokojí jen se základním dotazem *all*, či jeho obdobou *screen*. Přesto však specifikace nabízí značné množství dalších alternativ, které však nejsou zcela podporované napříč zařízeními. Využívání speciálních mediálních typů a výrazů je vhodné při potřebě vytváření unikátních webových stránek pro cílovou skupinu uživatelů. V praxi však není potřeba definovat vlastní styly pro každý typ zařízení, které je k v rámci mediálních dotazů k dispozici. Mnoho těchto zařízení se spokojí s výchozím nastavením kaskádových stylů.

Diskutovaným problémem je vytváření navigačního panelu stránek. Některé webové prezentace mají v hlavním panelu jen pár odkazů, jiné zde schovávají celou stromovou strukturu webu. Frameworky poskytují velmi praktické řešení pro definování tohoto panelu. To je však značně omezeno vzhledem a funkcionalitou. Při potřebě vytvářet unikátní navigační panely je vhodnější uchýlit se ke vlastnímu řešení, než se snažit modifikovat již hotový panel. Přesto, pokud v grafickém návrhu navržena klasická, všeobecně známá a na efekty nenáročná navigační lišta, je celkem praktické využít již hotovou komponentu. Z testovaných frameworků vychází lépe *Navbar* z Bootstrapu, který oproti Foundationu nevykazuje žádné chyby v zobrazení. Velmi obdobně je na tom i obrázkový slider, který je vhodné vytvářet podle požadavků daného projektu.

Při používání obrázků je nutné se zamyslet nad datovou náročností používaných dat. Dodnes využívaná metoda s využíváním velkého obrázku pro všechna rozlišení a jeho relativní přizpůsobování obsahu se stále jeví jako nejlepší alternativa pro zobrazování obrazových dat. V případě nutnosti omezit datovou náročnost se však vhodněji jeví element *picture*, který lépe reaguje na změny rozlišení zařízení. Tato vlastnost však může být jen důsledkem nedostatečné podpory v prohlížečích.

Při tvorbě webových formulářů je zcela vhodné využívat nové možnosti jazyka HTML5, který poskytuje velmi užitečné typy formulářových prvků, jež uživateli pro typ vkládaných dat zobrazí vhodnou klávesnici. Velkou výhodou této nové technologie je i automatické ověřování vložených dat do polí. Není tak nutné následně doplňovat stránky o skripty, které by ověřovaly obsah textových polí. Další nespornou výhodou je zpětná

kompatibilita se staršími prohlížeči, které v momentě nedostatečné podpory poskytnou uživateli výchozí typ textového pole.

Při práci s tabulkami bylo zjištěno, že možnosti tohoto elementu jsou velmi omezené. Nikoli však specifikací jazyka, ale samotnou podstatou tohoto prvku. Všechny 3 zmíněné přístupy využívají stejný princip pro práci s responsivními tabulkami a je tedy jedno, jakou alternativu vývojáři zvolí. Obdobně je na tom i skrývání a zobrazování prvků na stránce.

Při testování samotných přístupů byl zjištěn značný nedostatek ve funkcionalitě některých prvků frameworku Foundation 6, který vykazuje mnoho chyb v zobrazení. Některé tyto nedostatky lze vyřešit použitím starší verze frameworku, případně doplněním několika řádků zdrojového kódu. Z toho důvodu je však celkově framework Foundation 6 nevhodný pro širší používání na projektech.

Naopak framework Bootstrap je v mnoha ohledech velmi spolehlivým nástrojem pro tvorbu responsivních webových stránek. I zde je však značné množství nedostatků. Především značná nutnost předělávání již hotových kaskádových stylů. V omezené míře je však velmi praktické ho využívat na některé části webu.

5.2 Přínosy responsivního webu

V dnešní době je využívání responsivních přístupů k webovým prezentacím neodmyslitelnou součástí tvorby stránek. Stále více uživatelů prohlíží své oblíbené stránky na různých zařízeních a různých místech. Díky tomu jim webové stránky optimalizované pro většinu dostupných zařízení umožní maximální požitek. V dřívější době bylo značně problematické dosáhnout požadovaného obsahu na malých displejích a uživatelé tak ztráceli zájem o návštěvu stránek v momentě, kdy nebyli u svého počítače.

Díky responsivním stránkám však není problém dosáhnout jakéhokoliv obsahu stránek na všech zařízeních a umožnit tak maximální množství konverzí. Díky tomu mohou majitelům stránek narůstat tržby.

Velkým odvětvím moderního marketingu se stal tak zvaný *Online marketing*, který vedle různých cílených reklam a sběru informací o uživatelích pracuje také s možnostmi responsivního webu. Ze statistik společnosti Smart Insights (Bosomworth, 2016) je patrné, že velké procento uživatelů nalezne požadovaný obsah na svém mobilním zařízení,

samotnou konverzi však následně provede na stolním počítači. I z toho důvodu je nutné poskytnout uživatelům co nejlepší přístup k cílovému obsahu na mobilním zařízení.

5.3 Budoucnost responsivního webu

Valná většina nových možností pro tvorbu responsivních webových stránek se potýká s nedostatečnou podporou v prohlížečích. Lze však očekávat, že v brzké době, především kvůli snaze Microsoftu dostat svůj operační systém Windows 10 na většinu stolních počítačů, budou možnosti tvorby responsivních webových stránek značně rozšířeny.

Největší změna je očekávána u přístupu k rozvržení celého obsahu webové stránky a jeho následné přeskládávání napříč zařízeními. Zastaralá technologie s obtékáním jednotlivých elementů kolem sebe bude v brzké době pravděpodobně nahrazena takzvaným *Flexboxem*, který poskytuje podstatně lepší využívání prostoru obalového elementu.

6 Závěr

Nároky uživatelů na webové stránky jsou obrovské. Vytváření webu, který bude přístupný na všech zařízeních, je v dnešní době téměř nutností. Většina uživatelů prohlíží web na různých zařízeních, ať už jsou to chytré telefony, tablety, či různě velké notebooky, přičemž k internetu přistupuje v různých situacích, například z domova, v práci, či na cestách. Z mnoha hledisek je však nutné znát cílovou skupinu konkrétní webové stránky a té přizpůsobit samotný obsah.

Jednotlivé postupy pro tvorbu responsivního designu poskytují množství různých řešení, jak vytvořit stránky, přizpůsobující se většině dostupných zařízení. Nelze však jednoznačně určit, který z těchto postupů je tím nejlepším, protože každý se hodí pro jiný typ projektu.

Samotné postupy tvorby responsivního designu byly v praktické části rozděleny na několik základních obvykle používaných prvků webových stránek. Na základě jejich testování funkcionality, náročnosti na realizaci a podporu v prohlížečích bylo vzneseno doporučení k jejich praktickému využití.

Obecně se dá z výsledků praktické části říci, že většina nových technologií je značně náročnější na samotnou realizaci z důvodu jejich složitosti na zápis, což může zvýšit náklady na tvorbu stránek samotných. Naopak jejich výsledný efekt může vést k větší spokojenosti uživatelů a zvýšit tak množství samotných konverzí realizovaných na stránkách. Značnou minimalizaci nákladů poskytují i frontendové frameworky, které je však vhodné využívat v kombinaci s vlastním přístupem.

Závěrem bylo vzneseno celkové doporučení pro tvorbu responsivního designu. To je vždy závislé na konkrétním projektu a cílové skupině dané webové prezentace. Hlavní myšlenkou je tvořit stránky pro uživatele a situace, ve kterých se budou nalézat, než designování pro k zobrazení užitému zařízení. Uživatel stránek a jeho spokojenost jsou vždy na prvním místě.

Seznam zkratek

FUP - Fair Usage Policy

HTML – HyperText Markup Language

CSS – Kaskádové styly (Cascading Style Sheets)

JS – JavaScript

DPI – Dot Per Inch

PPI – Pixel Per Inch

DPR – Device Pixel Ratio

DIP – Density-Indepentent Pixel

PX – Pixel

WCAG – Web Content Accessibility Guidelines

SVG - Scalable Vector Graphics

CDN – Content Delivery Network

WYSIWYG - What You See Is What You Get

Citovaná literatura

Andrew, Rachel. 2015. Grid by example. *Grid by Example.* [Online] 2015. [Citace: 8. 11 2015.] http://gridbyexample.com/.

Apple. 2015. Apple iPhone. *Apple.* [Online] Apple Inc., 2015. http://www.apple.com/.

Bootstrap, Twitter. 2015. Bootstrap. *http://getbootstrap.com/.* [Online] Twitter, 2015. [Citace: 5. 5 2015.] http://getbootstrap.com/.

Bosomworth, Danyl. 2016. Mobile Marketing Statistics compilation. *Smart Insigts.* [Online] Smart Insights Ltd, 2016. [Citace: 2. 3 2015.] http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/.

Castro, Elizabeth a Hyslop, Bruce. 2012. *HTML5 a CSS3.* Brno : Computer Press, 2012. ISBN 978-80-251-3733-8.

Coyier, Chris. 2015. A Complete Guide to Flexbox. *CSS Tricks.* [Online] 8 2015. https://css-tricks.com/snippets/css/a-guide-to-flexbox/.

Craig Sharkie, Andrew Fisher. 2015. *Rresponsivní webdesign okamžitě.* Brno : Computer Press, 2015. 978-80-251-4384-1.

ČR, Parlament. 2000 - 2009. Úplné znění zákona. *Ministerstvo vnitra České republiky.* [Online] 2000 - 2009. [Citace: 11. 12 2015.] http://www.pristupnost.cz/pristupnost-webu-statni-spravy/zakon-365-2000-sb-o-informacnich-systemech-verejne-spravy/.

Depot, Web Designer. 2010-2016. *Web Designer Depot.* [Online] 2010-2016. http://www.webdesignerdepot.com.

Deveria, Alexis. 2016. Can I Use. *Can I Use.* [Online] 2016. [Citace: 15. 1 2016.] http://caniuse.com/.

Dobrý web, s. r. o. 2009. Přístupnost.cz. *Jak tvořit přístupný web.* [Online] Dobrý web, s. r. o., 2009. [Citace: 12. 12 2015.] http://www.pristupnost.cz/jak-tvorit-pristupny-web/.

Gaebel, Dennis. 2014. A Complete Guide to Grid. *CSS Tricks.* [Online] 14. 8 2014. [Citace: 11. 8 2015.] https://css-tricks.com/snippets/css/complete-guide-grid/.

Gasston, Peter. 2015. *Moderní web, perfektní stránky v každém prohlížeči a na každém zařízení.* Brno : Computer Press, 2015. 978-80-251-4345-2.

Gillenwater, Zoe Mickley. 2011. Essential considerations for crafting quality media queries. *Zomigi.* [Online] 21. 10 2011. [Citace: 22. 11 2015.] http://zomigi.com/blog/essential-considerations-for-crafting-quality-media-queries/.

Hansson, David Heinemeier. 2006. *Future Of Web Hansson, RailConf 2006.* [https://www.youtube.com/watch?v=GFhoSMD6idk] 2006.

Hogan, Brian. 2012. *HTML5 a CSS3 - výukový kurz webového vývojáře.* Brno : Computer Press, 2012. ISBN 978-80-251-3576-1.

Chapman, Cameron. 2010. Single-Page Websites: Examples and Good Practices. *Noupe.* [Online] 2010. [Citace: 7. 11 2015.] http://www.noupe.com/design/single-page-websites-examples-and-good-practices.html.

Jehl, Scott. 2015. Picturefill. *GitHub.* [Online] 2015. http://scottjehl.github.io/picturefill/.

Kadlec, Tim. 2014. *Responsivní design - profesionálně.* Brno : Zoner Press, 2014. 978-80-7413-280-3.

Lillevälja, Jaan-Matti. 2012. 19 Things We Can Learn From Numerous Heatmap Tests. *Conversion XL.* [Online] Conversion XL, 2012. http://conversionxl.com/19-things-we-can-learn-from-numerous-heatmap-tests/.

Love, One Page. 2014. What (exactly) is a One Page Website? *One page love.* [Online] 28. 10 2014. [Citace: 8. 11 2015.] https://onepagelove.com/what-exactly-is-a-one-page-website.

Lubbers, Peter, Albers, Brian a Salim, Frank. 2011. *HTML 5 - Programujeme moderní webové aplikca.* Brno : Computer Press, 2011. ISBN 978-80-251-3539-6.

Malánek, Michal. 2010. Kontrolní seznam pro WCAG 2.0. *Blind Friendly.* [Online] 31. 10 2010. [Citace: 2015. 10 15.] http://blindfriendly.cz/wcag20checklist/.

Marcotte, Ethan. 2010. Responisve web design. *A list apart.* [Online] 5 2010. http://alistapart.com/article/responsive-web-design. 1534-0295.

Market, Net. 2016. Mobile/Tablet Top Browser Share Trend. *Net Market.* [Online] 2016. https://www.netmarketshare.com/browser-market-share.aspx?qprid=1&qpcustomb=1.

Michále, Martin. 2015. Co je to „Mobile First“? Ale doopravdy. *Vzhůru Dolů.* [Online] 16. 12 2015. [Citace: 15. 1 2016.] http://www.vzhurudolu.cz/prirucka/mobile-first.

Michálek, Martin. 2015. Co je to „Mobile First“? Ale doopravdy. *Vzhůru Dolů.* [Online] 16. 12 2015. [Citace: 15. 1 2016.] http://www.vzhurudolu.cz/prirucka/mobile-first.

—. 2013. K čemu je dobrý Bootstrap a frontend frameworky? *Zdroják.cz.* [Online] 6. 12 2013. [Citace: 14. 1 2016.] https://www.zdrojak.cz/clanky/k-cemu-je-dobry-bootstrap-frontend-frameworky/.

—. 2015. Vzhůru dolů. *Vzhůru dolů.* [Online] 2015. [Citace: 14. 1 2016.] http://www.vzhurudolu.cz.

Pavlíček, Mgr. Radek. 2009. Ministerstvo vnitra České republiky. *Přístupný web a jak se vyvarovat chyb.* [Online] 2009. http://www.mvcr.cz/clanek/pristupny-web-a-jak-se-vyvarovat-chyb.aspx.

Pohanka, Pavel. 2015. Internet věcí. *Pavel Pohanka.* [Online] 2015. http://i2ot.eu/internet-of-things/.

Řezáč, Jan. 2012. *Jan Řezáč - Úvod do webdesignu.* [Vimeo] Brno : Kabinet informačních studií a knihovnictví FF MU Brno, 2012.

—. 2014. *Web ostrý jako břitva.* Jihlava : Baroque Partners, 2014. ISBN 978-80-87923-01-6.

Salminen, Viljami. 2015. ResponsiveSlides.js v1.54. *ResponsiveSlides.com.* [Online] 2015. http://www.responsiveslides.com.

Salvet, Pavel. 2009. Seznámení s medial queries v CSS 3. *Interval.* [Online] 6 2009. https://www.interval.cz/clanky/seznameni-s-media-queries-v-css-3/.

Schepers, Doug. 2013. Touch Events. *W3C.* [Online] 10 2013. http://www.w3.org/TR/touch-events/.

Schools, W3. 2015. CSS Media Types. *W3Schools.* [Online] 2015. http://www.w3schools.com/css/css_mediatypes.asp.

Snook, Jonathan. 2015. SIZING WITH CSS3'S VW AND VH UNITS. *Snook.ca.* [Online] 20. 8 2015. [Citace: 13. 1 2016.] http://snook.ca/archives/html_and_css/vm-vh-units.

Souček, Jiří. 2015. 1366×768 převálcovalo všechna nižší rozlišení dohromady, dominuje internetu. *diit.cz.* [Online] Aira GROUP, 27. 2 2015. http://diit.cz/clanek/nejrozsirenejsi-rozliseni-1366x768. 1213-2225.

Stojanov, Dimitar. 2015. A Visual Guide to CSS3 Flexbox Properties. *Scotch.* [Online] 4 2015. https://scotch.io/tutorials/a-visual-guide-to-css3-flexbox-properties.

Sucharda, Lukáš. 2013. Responsive Web Design (RWD). *BloFG.* [Online] 5 2013. http://blog.fg.cz/clanky/26/responsive-web-design-rwd.

Špinar, David a Pavlíček, Radek. 2006. Pravidla tvorby přístupného webu. *Pravidla přístupnosti.* [Online] 2006. http://www.pravidla-pristupnosti.cz/. YA512006003.

Špinar, David. 2000. Na webu. *Pravidla tvorby přístupného webu.* [Online] 2000. http://pristupnost.nawebu.cz/texty/pravidla-standardy.php?full#rule1.

Tichá, Petra a Dlouhý, Michal. 2014. Datové formáty videa, jejich specifikace a možnosti využití. *Multimediální technologie (UMT).* [Online] 19. 11 2014. https://umt.wikispaces.com/Datov%C3%A9+form%C3%A1ty+videa,+jejich+specifikace+a+mo%C5%BEnosti+vyu%C5%BEit%C3%AD.

Vracovský, Václav. 2015. Jak používáme Bootstrap. *Medio Blog.* [Online] Medio Interactive, 13. 1 2015. [Citace: 14. 1 2016.] http://blog.medio.cz/jak-pouzivame-bootstrap.

W3C. 2008. Web Content Accessibility Guidelines (WCAG) 2.0. *W3.* [Online] W3, 11. 12 2008. [Citace: 13. 10 2015.] https://www.w3.org/TR/WCAG20/.

W3Schools. 2015. CSS Fonts. *W3Schools.* [Online] 2015. http://www.w3schools.com/css/css_font.asp.

—. 1999-2016. CSS3 box-sizing Property. *W3Schools.* [Online] 1999-2016. [Citace: 27. 12 2015.] http://www.w3schools.com/cssref/css3_pr_box-sizing.asp.

WebAim. 2015. WebAIM's WCAG 2.0 Checklist. *WebAIM's.* [Online] 2015. [Citace: 1. 1 2016.] http://webaim.org/standards/wcag/checklist.

Wroblewski, Luke. 2011. *Mobile first.* New York : A Book Apart, 2011. 978-1-937557-02-7.

Zurb. 2016. Foundation for Sites. *Foundation Zurb.* [Online] 2016. [Citace: 15. 1 2016.] http://foundation.zurb.com/sites/docs/.

Seznam obrázků

Seznam tabulek

Přílohy

I Webové prezentace

Webové prezentace vytvářené v „*Praktické části práce*“ jsou k dispozici online na adresách:

- Varianta webové prezentace bez použití frameworku
  - http://dp-clear.lukasmodry.cz/
- Varianta webové prezentace s frameworkem Bootstrap 3.3.6
  - http://dp-bootstrap.lukasmodry.cz/
- Varianta webové prezentace s frameworkem Foundation 6
  - http://dp-foundation.lukasmodry.cz/