Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Bakalářská práce

Responzivní web

Štěpán Farka

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Štěpán Farka

Informatika

Název práce

Responzivní web

Název anglicky

Responsive website

Cíle práce

Bakalářská práce je tématicky zaměřena na problematiku tvorby responzivních webů.
Hlavním cílem práce je analýza nástrojů a způsobů pro tvorbu responzivních webů s následnou implementací webové stránky.
Dílčí cíle práce jsou:
- vypracování přehledu používání webových stránek na mobilních zařízení,
- komparace nástrojů Bootstrap a Tailwind CSS,
- vypracování přehledu použitelnosti media a container queries,
- komparace přístupů pro tvorbu responizivního webu.

Metodika

Metodika řešené problematiky bakalářské práce je založena na studiu a analýze odborných informačních zdrojů. Vlastní práce spočívá v porovnání metod pro tvorbu responzivních webů s následnou aplikací vybraných metod na reálné webové stránce. Na základě syntézy teoretických poznatků a výsledků praktické části budou formulovány závěry bakalářské práce.

Doporučený rozsah práce

40 – 50 stran textu.

Klíčová slova

HTML, CSS, Tailwind CSS, Bootstrap, Flex, Grid, Responzivita

Doporučené zdroje informací

FRAIN, Ben. Responsive Web Design with HTML5 and CSS: Build future-proof responsive websites using the latest HTML5 and CSS techniques. 4th Edition. Packt Publishing, 2022. ISBN 978-1803242712.

GERCHEV, Ivaylo. Tailwind CSS. SitePoint, 2022. ISBN 978-1925836516.

MICHÁLEK, Martin. CSS: Moderní layout. Martin Michálek – VzhůruDolů.cz, 2022. ISBN 978-80-88253-07-5.

MICHÁLEK, Martin. Vzhůru do (responzivního) webdesignu. Martin Michálek – VzhůruDolů.cz, 2018. ISBN 978-80-88253-00-6.

WROBLEWSKI, Luke. Mobile First. A Book Apart, 2011. ISBN 978-1937557027.

Předběžný termín obhajoby

2023/24 LS – PEF

Vedoucí práce

doc. Ing. Pavel Šimek, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 4. 7. 2023

doc. Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 3. 11. 2023

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 12. 03. 2024

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci „Responzivní web“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 12.3.2024 _____

Poděkování

Rád bych touto cestou poděkoval panu doc. Ing. Pavlovi Šimkovi za vedení bakalářské práce, cenné rady a věcné připomínky při konzultacích.

Responzivní web

Abstrakt

Bakalářská práce je zaměřena na problematiku tvorby responzivních webových stránek. Hlavním cílem práce je analýza nástrojů a způsobů, které se používají k tvorbě responzivních návrhů.

Teoretická část obsahuje literární rešerši odborných informačních zdrojů, kde jsou představeny moduly, jako flexbox a grid, které se v současnosti používají při implementaci responzivních webových stránek. Jsou představeny a pospány dílčí vlastnosti jednotlivých modulů. Dále se teoretická část zaměřuje na představení techniky Container Queries. Součástí teoretické části je seznámení s CSS frameworky Bootstrap a Tailwind CSS a popsání způsobu vykreslení webové stránky.

Praktická část je zaměřena na tvorbu responzivních komponent s následnou implementací webové stránky. Jednotlivé komponenty jsou nejprve navrženy a následně implementovány. Implementace jsou porovnány srovnáním se stanovenými kritérii. Pomocí analýzy SWOT je zhodnocena implementace techniky Container Queries. Z porovnání a analýzy jsou formulovány výsledky.

Klíčová slova: HTML, CSS, Tailwind CSS, Bootstrap, Flex, Grid, Responzivita

Responsive website

Abstract

The bachelor thesis is focused on the issue of creating responsive websites. The main goal of the thesis is to analyse the tools and methods that are used to create responsive designs.

The theoretical part includes a literature search of professional information sources, where modules such as flexbox and grid, which are currently used in the implementation of responsive websites, are introduced. The partial properties of each module are introduced and discussed. Furthermore, the theoretical part focuses on the introduction of the Container Queries technique. The theoretical part includes an introduction to the Bootstrap and Tailwind CSS frameworks and a description of how to render a web page.

The practical part focuses on the creation of responsive components with subsequent implementation of a web page. The individual components are first designed and then implemented. The implementations are compared against the established criteria. Using SWOT analysis, the implementation of the Container Queries technique is evaluated. From the comparison and analysis, results are formulated.

Keywords: HTML, CSS, Tailwind CSS, Bootstrap, Flex, Grid, Responsivity

Obsah

1 Úvod

Responzivita je dnes považována za základní stavební kámen každé webové stránky. S rostoucím počtem mobilních telefonů, přichází očekávání, že webové stránky budou poskytovat uživatelsky přívětivé chování i na kompaktních zařízeních. K tvorbě takových webů existuje řada nástrojů a technik, které nám umožňují vytvářet rozdílné pohledy v závislosti na velikosti rozlišení.

Responzivní design je přístup, který se vyvinul v posledním desetiletí. V počátku webové stránky obsahovaly pouze textové řetězce připomínající dnešní aplikaci Word. Později se přešlo na strukturování obsahu pomocí tabulek, avšak tento přístup se ukázal jako nevhodný, jelikož tabulky byly určeny pro prezentaci dat, nikoli uspořádání obsahu stránky. Výsledkem takové stránky byl špatně čitelný kód, který obsahoval složité zanoření prvků.

S příchodem kaskádních stylů se přešlo na tvorbu rozložení pomocí plovoucích prvků, jednalo se o jednu z prvních technik, která umožňovala tvorbu rozložení pro různá zařízení. Jednotlivé prvky byly umístěny vedle sebe nebo pod sebou v závislosti na velikosti dostupného místa.

S narůstající komplexitou webových návrhů vznikly nové moduly pro tvorbu responzivních webových stránek.

Hlavním cílem této práce je analýza nástrojů a způsobů pro tvorbu responzivních webů s následnou implementací webové stránky. V práci jsou představeny moduly a techniky, které nabízejí intuitivnější syntaxi a lepší kontrolu nad uspořádáním a umožňují například snadnější práci s přesouváním prvků na stránce.

Mezi dílčí cíle práce náleží vypracování přehledu použitelnosti na mobilních zařízeních, komparace nástrojů Bootstrap a Tailwind CSS, vypracování přehledu použitelnosti Container Queries a Media Queries a komparace přístupů pro tvorbu responzivních webových stránek.

Praktická část práce je zaměřena na vytvoření návrhu a implementace responzivních komponent s následnou analýzou nebo porovnáním. Dále je provedena integrace jednotlivých komponent do kompletní responzivní webové stránky. Na základě zjištěných skutečností jsou formulovány výsledky.

2 Cíl práce a metodika

2.1 Cíl práce

Bakalářská práce je tématicky zaměřena na problematiku tvorby responzivních webů. Hlavním cílem práce je analýza nástrojů a způsobů pro tvorbu responzivních webů s následnou implementací webové stránky. Dílčí cíle práce jsou:

- vypracování přehledu používání webových stránek na mobilních zařízení,
- komparace nástrojů Bootstrap a Tailwind CSS,
- vypracování přehledu použitelnosti Media a Container Queries,
- komparace přístupů pro tvorbu responzivního webu

2.2 Metodika

Metodika řešené problematiky bakalářské práce je založena na studiu a analýze odborných informačních zdrojů. Zpracování literární rešerše přináší rozšíření teoretického poznání. Stanoviska jednotlivých autorů jsou vzájemně provázaná a získané poznatky jsou aplikovány v rámci praktické části.

Teoretická část je zaměřena na představením přístupů a principů k dosažení responzivního rozložení a aplikování nástrojů k tvorbě responzivních webových stránek.

V praktické části, na základě poznatků ze zpracování informačních zdrojů, jsou porovnány metody a techniky pro tvorbu responzivních webových stránek. Je proveden návrh responzivních komponent s následnou implementací. Porovnání je prováděno pomocí srovnání s definovanými hodnotícími kritérii a analýza technik je provedena pomocí SWOT analýzy. Tyto metody umožní zhodnocení sledovaných metod a technik pro tvorbu responzivních webových stránek a umožní objektivní posouzení efektivity a vhodnosti pro praktické použití.

Na základě syntézy teoretických poznatků a výsledků praktické části je možné aplikovat získané znalosti při vývoji responzivní webové stránky v praxi. Z výsledků a zhodnocení vlastní práce jsou formulovány závěry bakalářské práce.

3 Teoretická východiska

3.1 Responzivní design

Termín responzivní design byl popularizován článkem, který napsal v roce 2010 vývojář Ethan Marcotte. Úkolem responzivního designu je zajistit, aby webová stránka působila esteticky příjemně a byla funkční bez ohledu na rozměry obrazovky používaného zařízení. Před příchodem responzivního designu vytvářeli webový designéři webové stránky podle principu pixel-perfect, který pojímal webovou stránku jako například stránku v časopise. Je přesně stanovená šířka stránky a jednotlivých prvků. Webová stránka však nefunguje jako papír, ale zobrazí se ve webovém prohlížeči, který uživateli umožňuje dynamicky měnit šířku stránky. [1]

Responzivní design využívá kombinaci flexibilních mřížek, obrázků a mediálních dotazů k vytvoření responzivního rozvržení. Tímto způsobem se obsah automaticky přizpůsobuje zařízením uživatelů. [2]

3.1.1 Výhody responzivního designu

- Webové stránky vypadají a fungují dobře, což zlepšuje uživatelský zážitek
- Responzivní design nevyžaduje duplikaci kódu, což může zlepšit SEO optimalizaci
- Tvorba responzivních webových stránek je nákladově efektivnější, než vytváření samostatných verzí pro různá zařízení

Tato část je zpracovaná podle [2].

3.1.2 Nevýhody responzivního designu

- Responzivní design vyžaduje pečlivé rozplánovaní provedení webových stránek
- Mobilní verze obsahuje styly, které se aplikují pouze na větších zařízeních, což vede k pomalejšímu načítání webových stránek
- Responzivní design omezuje vzhled jednotlivých prvků, prvky by měly být standardizovanější a jednotnější

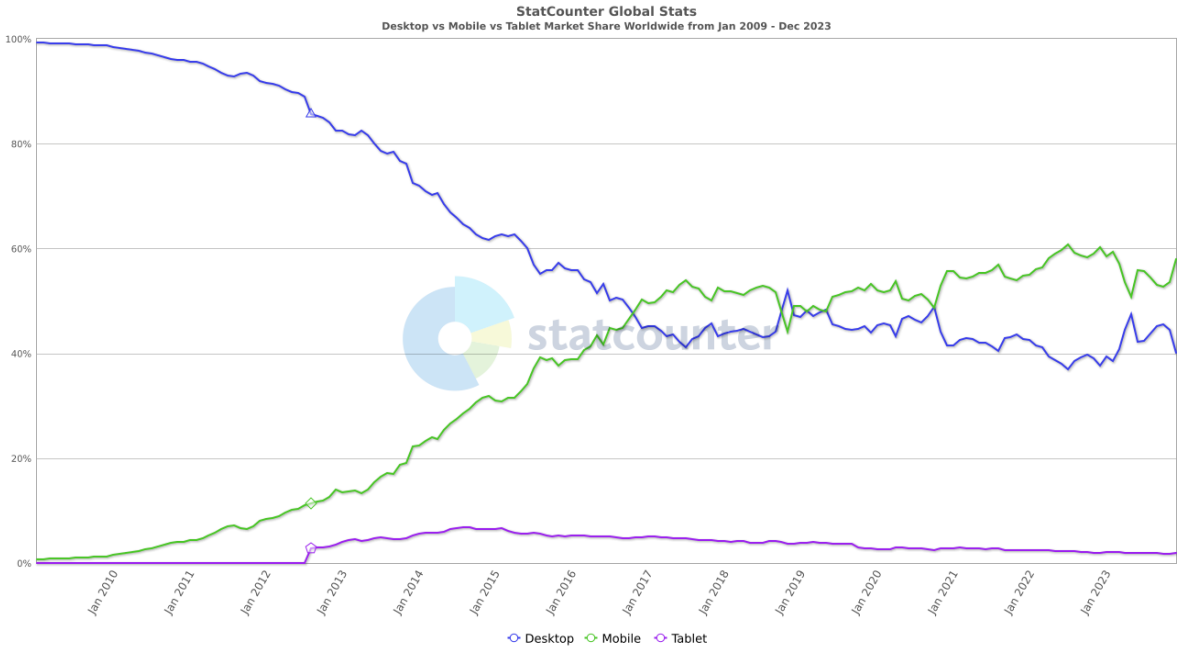
Tato část je zpracovaná podle [2].

3.2 Používání webových stránek na mobilu

Mobilní zařízení se staly každodenní součástí života milionů lidí. Zařízení s podporou webového prohlížeče, jako jsou například mobilní telefony nebo tablety, po

celém světě představují základní nástroj komunikace, zábavy a zdroj informací. V roce 2022 činil počet unikátních uživatelů, využívající internet v mobilních zařízeních, pět miliard. To představuje 60 % celosvětové lidské internetové populace. Předpokládá se, že počet mobilních zařízení a využívání internetu bude v budoucnu nadále růst, jelikož jsou technologie cenově dostupnější. Tento vzestupný trend v přijetí mobilního internetu je zvláště patrný v rozvíjejících se digitálních trzích. Zde jsou mobilní zařízení hlavním prostředkem přístupu k internetu. [3]

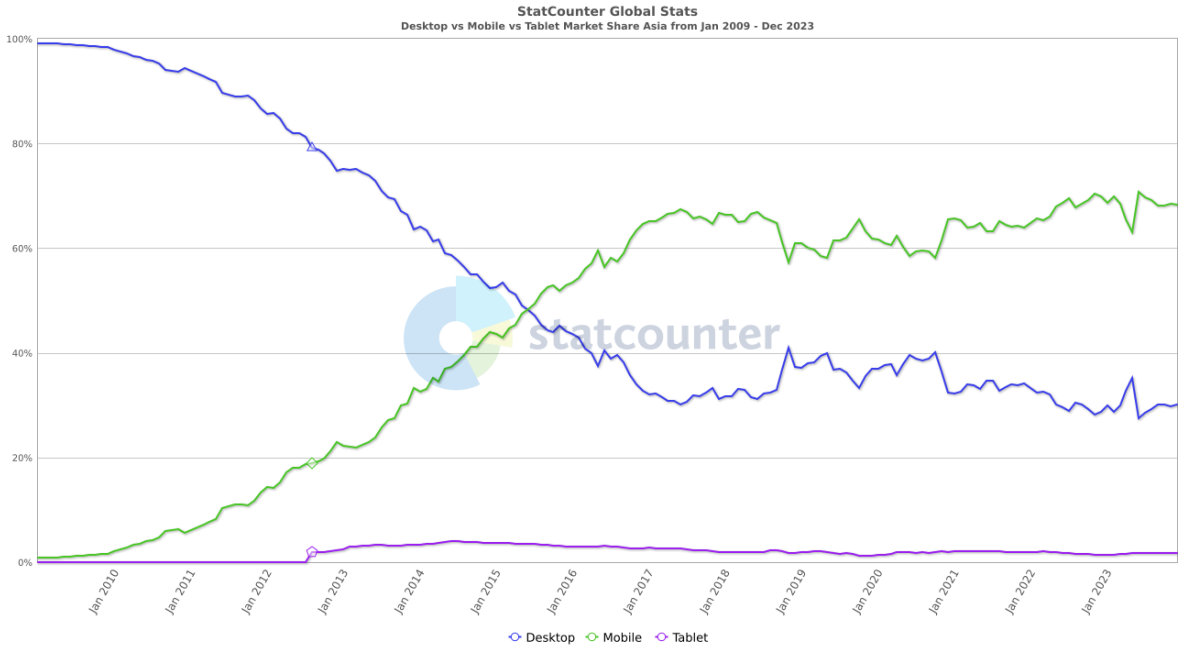
Obrázek 1 Graf zachycující celosvětový poměr zařízení na internetu 2009–2023



Zdroj: [4]

Mobilní zařízení na internetu dnes reprezentují téměř 60 % celkového provozu. U trhů zaměřených na mobilní zařízení, jako jsou například Asie nebo Afrika, je podíl na zobrazení webových stránek z mobilních zařízení ještě větší. [3]

Obrázek 2 Graf zachycující poměr zařízení na internetu 2009–2023 v Asii



Zdroj: [5]

3.3 Přístupy pro tvorbu responzivního webu

K dosažení responzivního designu existuje několik technik, které se aplikují při tvorbě webových stránek. Jednou z populárních strategií je mobile first, která z pohledu důležitosti staví mobilní zařízení na úroveň desktopového počítače. [6]

Dalšími přístupy jsou definice zlomových bodů, které přesně specifikují změnu vzhledu při určitém rozlišení nebo využívání flexibilních jednotek typu *em*, *rem*, které umožňují pružné uspořádání na základě velikosti obrazovky. [7]

3.3.1 Mobile first

Myšlenka vytvářet webové rozložení primárně z mobilních zařízeních vzešla z faktu, že mobily budou používanější než desktop. Pokud návrh počítá primárně s desktopovým rozlišením, často se tento návrh špatně přenáší na mobilní zařízení, jelikož desktop poskytuje více místa. [6]

Designéři připravují návrhy webové stránky primárně pro mobilní uživatele. Celý proces návrhu probíhá s ohledem na malou obrazovku, po níž následují upravené verze pro tablety a následně desktopové počítače nebo notebooky. Strategií pro tvorbu rozhraní je zaměřit se pouze na výstižný obsah, který by měl obsahovat jasné pořadí. Cílem je

uživateli poskytnout rozložení, ve kterém mu bude zřejmé, jakým způsobem se stránkou pracovat. Výzvy k akci by měli obsahovat konzistentní design. [8]

3.3.2 **Desktop first**

Desktop first je způsob, který upřednostňuje navrhování a kódování webových stránek pro větší obrazovky počítačů a notebooků. Taková řešení obvykle zahrnují rozvržení s více sloupci a pokročilé interakce, jako jsou efekty při najetí myši na určitý element. Obsahují prvky, které nemusí na mobilních zařízeních dobře fungovat, například vyskakovací okna. Designéři mají výhodu velkého prostoru pro návrh rozložení. [2]

3.3.3 **Zlomové body (Breakpoints)**

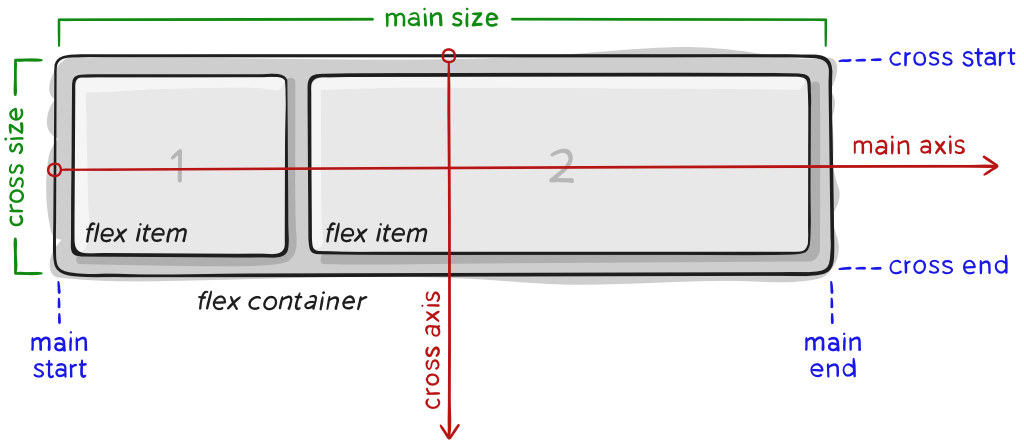
Zlomové body představují další způsob pro tvorbu responzivního designu, který je založen na změně rozložení v určitém bodě. Například pokud dojde příliš k zmenšení obsahu je potřeba zvážit změnu stylů. Tento způsob umožňuje se zaměřit na konkrétní velikosti obrazovek a vytvořit pro ně specifický design, avšak vzhledem k rozmanitosti dostupných zařízení je určení správného bodu zlomu velice náročné. V rámci zjednodušení obvykle designéři seskupují zařízení do určitých kategorií, kterými mohou být například mobilní telefony, tablety, notebooky nebo desktopové počítače. [9]

3.4 **Flexbox**

Myšlenkou flexboxu je poskytnou kontejneru možnost měnit velikost položek v závislosti na své velikosti, aby vyplnily co nejlépe volný prostor. Modul byl navržen jako jednorozměrný model rozložení, který pracuje v režimu řádku nebo sloupce. [10]

Vzhledem k tomu, že flexbox není jedna vlastnost, ale jedná se o celý modul, zahrnuje několik vlastností. Některé z nich poskytují možnosti pro nastavování kontejneru (rodičovského prvku, který v terminologii je označován jako flex container) a ostatní k nastavování vlastností u potomků (označovány jako flex items). Položky jsou umístěny buď podle hlavní osy zleva-doprava nebo podle příčné osy shora-dolů. Ve výchozím stavu jsou položky seřazeny podle hlavní osy. Nemusí se vždy jednat o horizontální směr, jelikož pomocí vlastnosti *flex-direction: column* lze změnit hlavní osu do vertikální polohy. [11]

Obrázek 3 Rozložení flexboxu



Zdroj: [11]

3.4.1 Historie

Obdobně jako všechny specifikace CSS prošel i flexbox velkým počtem změn, než se z něj stalo doporučení od konsorcia W3C. Flexbox byl experimentálně implementován do několika webových prohlížečů. V té době bylo běžnou praxí při vytváření experimentálních implementací používat předponu definovanou výrobcem. Předpony umožňovaly inženýrům webových prohlížečů a vývojářům webových aplikací testovat a zkoumat implementace specifikace, aniž by docházelo ke střetu s jinými implementacemi. Původním záměrem bylo nepoužívat experimentální implementace v kódu, nicméně prefixy se nakonec v produkčním kódu používaly a změny experimentální specifikace znamenaly, že lidé museli aktualizovat své webové prohlížeče, aby s novinkami drželi krok. V roce 2009 vypadala specifikace flexboxu zcela jinak. Pro vytvoření flexibilního kontejneru byla používána vlastnost *display: box* a existoval soubor vlastností s prefixem *box-**, které dělaly věci, které jsou známé z flexboxu dnes. Ve specifikaci se objevila verze, která syntaxi upravovala na *display: flexbox*. V konečném důsledku byla specifikace upravena tak, že se stanovilo pro vytvoření flexibilního kontejneru použití vlastnosti *display: flex*. Podpora prohlížečů pro novou verzi specifikace je od tohoto okamžiku velmi dobrá. [12]

3.4.2 Přehled vybraných vlastností flexboxu

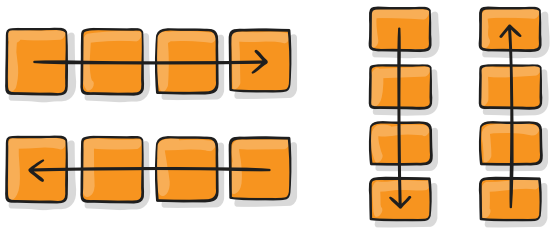
flex-direction

Vlastnost flex-direction se aplikuje na kontejner flexboxu a určuje způsob seřazení položek tím, že specifikuje hlavní osu flexboxu. Vlastnost může nabývat hodnot:

- row (jedná se o výchozí hodnotu, která definuje rozmístění položek v řádku)
- row-reverse (hodnota umožňující opačné pořadí nastavení položek v řádku)
- column (položky se skládají shora dolů, v rámci jednoho sloupce)
- column-reverse (položky se nacházejí v sloupci v opačném pořadí)

Tato část je zpracována podle [10].

Obrázek 4 Směry atributu flex-direction



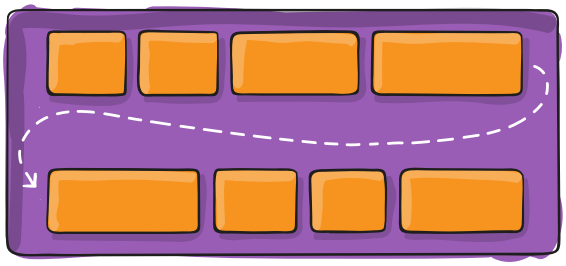
Zdroj: [11]

flex-wrap

Vlastnost flex-wrap slouží ke specifikování, zdali se položky mohou zalamovat na více řádků. Vlastnost nabízí hodnoty:

- nowrap (výchozí hodnota, která říká, že flexbox bude pouze jednořádkový)
- wrap (kontejner umožní položkám zalomení na další řádek)
- wrap-reverse (kontejner opět umožní zalomení s rozdílem opačného pořadí položek)

Obrázek 5 Vlastnost flex-wrap



Zdroj: [11]

Tato část je zpracovaná podle [13].

flex-flow

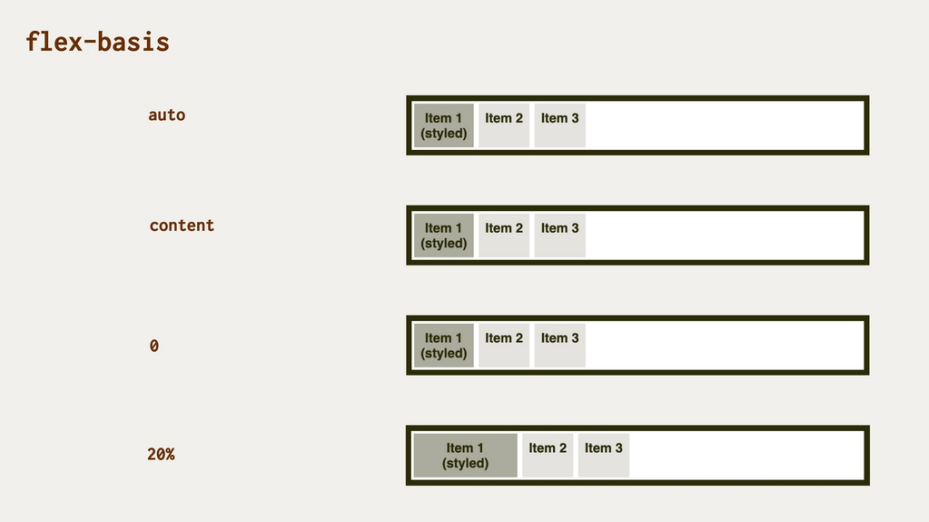
Vlastnost *flex-flow* představuje dílčí modul flexboxu, jedná se o zkratku pro vlastnosti *flex-direction* a *flex-wrap*. Zápis pro vlastnost je *flex-flow: <hodnota flex-direction> <hodnota flex-wrap>*. Vlastnosti *flex-direction* a *flex-wrap* používají jiná klíčová slovo, a proto je možno uvádět hodnoty v libovolném pořadí nebo jednu z nich zcela vynechat. [14]

flex-basis

Hodnota *flex-basis* určuje počáteční velikost prvku před rozdělením prostoru. Výchozí hodnota této vlastnosti je *auto*. Pokud je hodnota nastavena na *auto*, prohlížeč nejprve zkontroluje, zda je hlavní rozměr prvku nastaven na pevno (absolutně), což nastává, pokud je konkrétní šířka prvku například 200 px. V takovém případě bude 200 px sloužit jako základ pro flex-basis nad tímto prvkem. [15]

Pokud je prvek nastaven na automatickou velikost podle obsahu, bude automatická velikost aplikována na rozměry jeho obsahu. V této situaci je důležité znát maximální rozměry obsahu, jelikož flexbox vezme maximální rozměr jako flex-basis pro danou položku. [15]

Obrázek 6 Vlastnost flex-basis



Zdroj: [16]

flex-grow

Vlastnost *flex-grow* určuje faktor růstu, o kolik se prvek zvětší vzhledem k ostatním prvkům ve flex kontejneru při rozdělení zbývajícího místa.

Pokud mají všechny položky stejný faktor zvětšování (flex-grow), prostor mezi nimi bude rozdělen rovnoměrně. Vlastnost může nabývat poměrů například 1,2 nebo 88,100. [17]

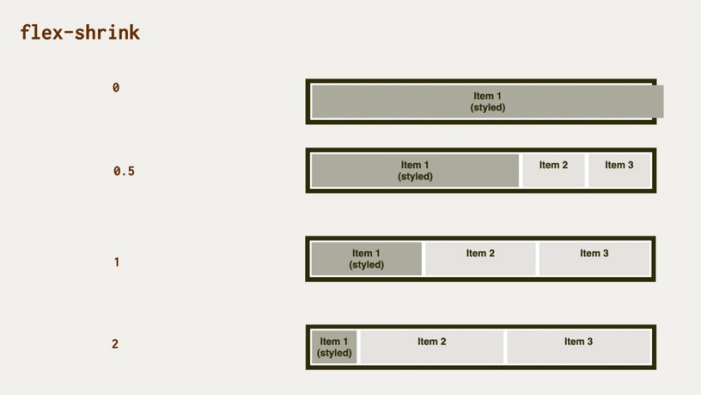
flex-shrink

Vlastnost *flex-shrink* udává, jakým způsobem se může položka zmenšovat, pokud v rodičovském kontejneru ubylo místo. Vlastnost může nabývat následujících hodnot:

- 1 - jedná se o výchozí stav, položka ubere o odpovídající část velikosti zmenšení kontejneru.
- Kladná čísla (0.5, 1, 2...) - položka v kontejneru si ze své velikosti ubere podíl zmenšení kontejneru odpovídající zvolené hodnotě.
- 0 - položka v kontejneru se nebude zmenšovat a bude mít stále stejnou velikost.

Tato část je zpracována podle [18].

Obrázek 7 Vlastnost flex-shrink



Zdroj: [18]

3.5 Grid

Grid představuje novější a robustnější nástroj pro tvorbu rozložení v kaskádových stylech. Jedná se o sadu vlastností pro tvorbu layoutu vsazeného do pravidelné mřížky. Výhoda gridu spočívá v možnosti definovat mřížku v obou směrech – v řádcích i sloupcích. V porovnání s flexboxem se jedná o vhodnější nástroj pro tvorbu komplexnějších návrhů. [13]

3.5.1 Podpora gridu

Podpora gridu je v moderních prohlížečích výborná. Výjimku představuje prohlížeč Internet Explorer 11 od společnosti Microsoft. Explorer a starší verze prohlížeče Edge,

podporují dnes již neplatnou verzi specifikace gridu. Jedná se o specifikaci, která má jinou syntaxi a nabízí podmnožinu dnešních vlastností. Dále nepodporuje například zarovnání prvků do mřížky nebo neumí vlastnost *grid-gap*. [19]

Obrázek 8 Podpora gridu v prohlížečích

Zdroj: [20]

3.5.2 Porovnání s flexboxem

Grid a flexbox nejsou zástupné moduly, k tvorbě responzivních layoutů je často vhodná kombinace těchto modulů. Grid je vhodnější pro dvourozměrné layouty, zatímco flexbox se více hodí na jednodimenzionální rozložení. Grid se častěji používá pro celkové rozložení stránky a flexbox pro dílčí komponenty. Modul grid je více zaměřený na layout princip *„grid in“*, kdy se obsah musí přizpůsobit mřížce. Flexbox je vhodnější na situace *„content out“*, kdy se layout přizpůsobuje svému obsahu. [19]

3.5.3 Speciální jednoty a funkce v gridu

Velikost prvků v mřížce lze definovat pomocí klasických délkových jednotek v CSS. Mřížkové rozložení zavádí i další jednotku, která pomáhá vytvářet flexibilní zobrazení. Jednotka *fr* představuje podíl dostupného prostoru v kontejneru mřížky. [21]

fit-content()

Funkce využívající dostupné místo. Nikdy méně, než je min-content a nikdy více než je max-content prvku.

minmax()

Funkce, která prvku nastavuje minimální a maximální délku, které může nabývat.

reapeat()

Funkce nabízí usnadnění zápisu. Prvním argumentem je počet, kolikrát se má druhý argument (vlastnost) zopakovat. Například místo zápisu *grid-template-columns: 1fr 1fr 1fr 1fr,* lze využít funkci repeat a zápis by vypadal *grid-template-columns: repeat(4, 1fr)*

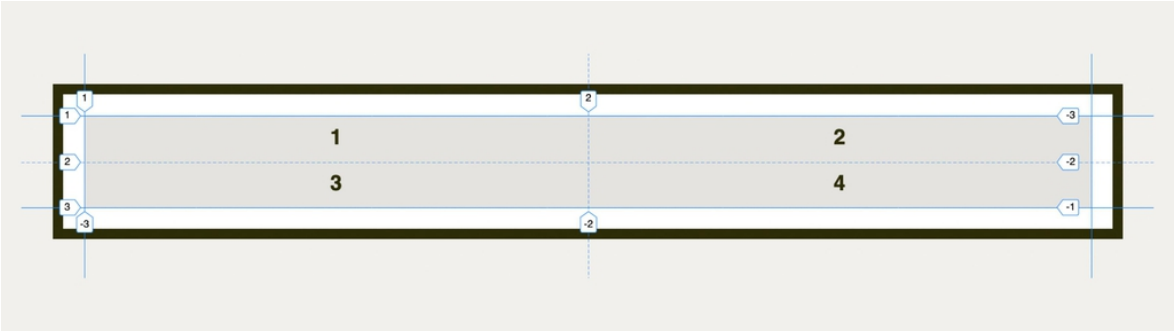
3.5.4 Přehled vybraných vlastností gridu

grid-template-columns a grid-template-rows

Definuje sloupce nebo řádky pomocí seznamu, který je oddělený mezerou. V gridu je možné pro definici řádků a sloupců používat všechny jednotky, které se v CSS nachází. [13]

Pro definování mřížky je nejprve potřeba zapnout mřížkové zobrazení pomocí vlastnosti *display: grid*. Následně se definují sloupce pomocí vlastnosti *grid-template-columns: 50% 50%.* To nastaví dva sloupce, který budou zabírat polovinu šířky svého rodičovského kontejneru. Definici řádků určuje vlastnost *grid-template-rows: auto auto*. Klíčové slovo *auto* v tomto případě znamená, že výška řádku bude odpovídat výšce obsahu. [22]

Obrázek 9 Vlastnost grid-template-columns a grid-template-rows



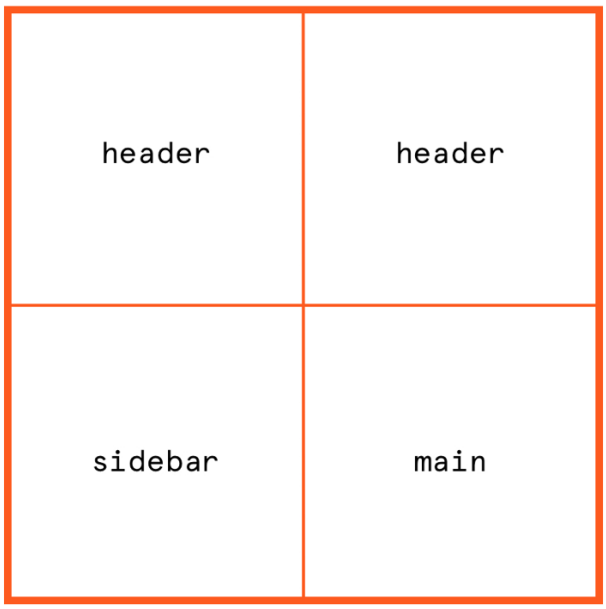
Zdroj: [22]

Pro větší přehlednost vlastnost nabízí možnost pojmenování linek, která se specifikuje pomocí hranatých závorek před velikostí sloupce či řádku. [13]

grid-template-areas

Vlastnost, která představuje možnost pojmenování oblastí definovaných gridem. Definuje oblasti, které následně lze použít u vlastnosti grid-area. Například pomocí vlastnosti *grid-template-areas: "header header" "sidebar main"* je definována mřížku dva na dva, kde horní řádek vymezuje oblast s názvem **header** a spodní řádek je rozdělen do oblastí **sidebar** a **main**. Pro přiřazení prvku do oblasti je nutno u prvku použít vlastnost *grid-area* s názvem oblasti. [23]

Obrázek 10 Vlastnost grid-template-areas

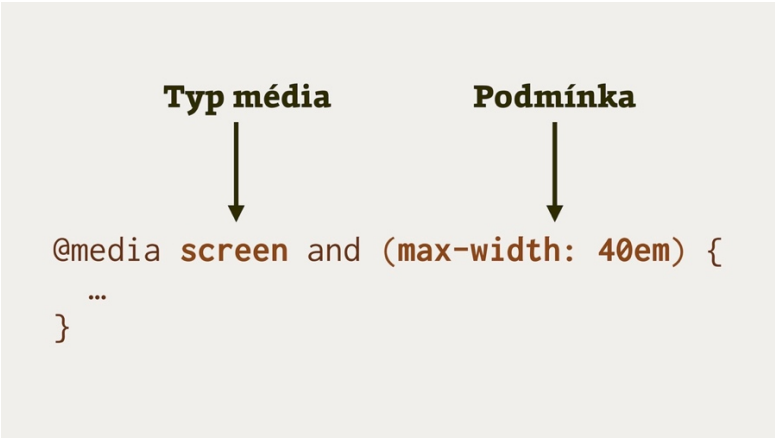


Zdroj: [23]

3.6 Media Queries

Media Queries představují podmínky umožňující aplikaci různých kaskádních stylů v závislosti na pravidlech. Dotaz na médium se skládá z typu média a podmínky, která obsahuje vlastnost média s hodnotou nebo rozsahem hodnot. [24]

Obrázek 11 Anatomie Media Query



Zdroj: [24]

3.6.1 Body zlomu

V responzivním designu představuje hodnota bodu zlomu vlastnost média. Jedná se o sadu hodnot pro web nebo systém designu. Hodnoty určují, kdy se mají aplikovat určité

styly. Například pokud bude použito menu v navigační liště, tak pomocí zlomového bodu lze nastavit jiné seskupení položek pro mobilní zařízení. [24]

V počátcích responzivních návrhů se mnoho designerů snažilo zaměřit na specifické velikosti obrazovek. Byly zveřejňovány seznamy velikostí obrazovek oblíbených telefonů a tabletů, aby bylo možné vytvářet návrhy na míru. V současné době existuje příliš mnoho různých velikostí. To znamená, že namísto zaměření se na konkrétní velikost obrazovky, je vhodnější optimalizovat návrh v místech, kde se začne nějakým způsobem porušovat. Například pokud se délka řádku stane příliš dlouhou a obsah v postranním panelu není již čitelný. To představuje bod, ve kterém je vhodné použít dotaz na média a změnit zobrazené rozložení webové stránky. Tyto body se označují za body zlomu. [25]

3.6.2 Logické operátory

Media Queries umožňují spojovat podmínky pomocí logického operátoru *and*. Operátor *or* nepodporují, místo něj se používá oddělovač čárka. Kombinace logických operátorů může například specifikovat kaskádní styly, které se mají aplikovat na určitém výseku layoutu. Příkladem je zápis, který definuje, že styly se budou aplikovat na všechny typy médií s šířkou od 30 em do 40 em. [26]

@media all and (min-width: 30em) and (max-width: 40em) { ... }

3.6.3 Detekce orientace zařízení

Jednou z dobře podporovaných funkcí Media Queries je orientace, která umožňuje testovat režim na výšku nebo na šířku. Výchozí nastavení zobrazení je na šířku a design, který v této orientaci funguje dobře, nemusí fungovat stejně dobře při zobrazení na telefonu či tabletu v režimu na výšku. Pro zobrazení na šířku je vlastnost *orientation* nastavena na hodnotu *landscape* a pro zobrazení na výšku na hodnotu *portrait*. [25]

3.6.4 Úprava vzhledu pro tisk

Webové stránky, které prodávají například vstupenky nebo nabízejí možnost stažení faktury, často nabízí optimalizovanou verzi webové stránky pro možnost tisku. U každodenního obsahu to není pravidlem. Pro úpravu vzhledu webových stránek k tisku je možné definovat CSS styly pomocí dotazu na média. Pro definování stylů pro tisk je potřeba vytvořit dotaz na média, kde argumentem je klíčové slovo *print*. Příklad média dotazu: *@media print { ... }* [27]

3.6.5 Detekce barevného režimu

Funkce *prefers-color-scheme* se používá k detekci, zdali uživatel preferuje tmavý nebo světlý režim zobrazení. Uživatel dává najevo své preference prostřednictvím nastavení operačního systému nebo nastavením uživatelského agenta. Pro nastavení vlastností CSS pomocí dotazu na média při tmavém barevném režimu se definuje dotaz následovně: *@media (prefers-color-scheme: dark) { ... }* [28]

3.7 Container Queries

Container Queries představují nový přístup v tvorbě responzivního designu webových stránek. V roce 2017 se této technologii říkalo „Element queries“, které přestavovaly dotazy na rozměry konkrétního prvku stránky. V té době bylo možné technologii emulovat pouze pomocí JavaScriptových knihoven. Lidé ze standardizační organizace W3C tehdy nad Container Queries přemýšleli, ale zdálo se jim, že je to příliš složité na implementaci v prohlížečích. Následně debata na mnoho let utichla a zůstalo jen u JavaScriptiových knihoven, které nikdy neposkytovaly dostatečnou rychlost na vykreslení stránky. V roce 2020 v prosinci přišla s novým návrhem implementace Miriam Szzanem, jejíchž návrh byl postaven i na mnoha letech práce i ostatních autorů. Návrh se skládal ze dvou kroků. Prvním byla definice kontejneru a druhým dotaz na samotný kontejner. [29]

Dotazy na kontejner umožňují použití stylů na základě velikosti kontejneru prvku. Například pokud má kontejner v okolním kontextu k dispozici méně místa, může skrýt určité prvky nebo například použít menší velikost písma. Kontejnerové dotazy jsou alternativou k mediálním dotazům, které aplikují styly na prvků v závislosti na velikosti zobrazení nebo jiných charakteristik zařízení. [30]

3.7.1 Deklarace kontejneru

Pro použití techniky Container Queries je potřeba nejprve definovat kontext kontejneru, na základě, kterého prohlížeč umožní se dotazovat na rozměry. K tomu je potřeba definovat vlastnost *container-type*, která může nabývat hodnot *size*, *inline-size* nebo *normal*.

- **size** – Dotaz bude založen na inline a blokových rozměrech kontejneru. Na kontejner se aplikuje rozložení, styl a omezení velikosti. U kontejneru se musí nastavit explicitně šířka a výška.

- **inline-size** – Dotaz bude založen na inline rozměrech kontejneru. Na kontejner se aplikuje rozložení, styl a omezení velikosti na základě viewportu.
- **normal** – Kontejner není kontejnerem pro dotazy na velikost kontejneru, ale zůstává kontejnerem pro dotazy na styl kontejneru

Tato část je zpracována podle [30].

3.7.2 Pojmenování kontejneru

Nastavení názvu kontejneru je nepovinné, avšak se doporučuje, jelikož snižuje chyby při práci s dlouhými CSS soubory. Pojmenování je citlivé na malá a velká písmena. Pro nastavení názvu se nastavuje vlastnost *container-name* s hodnotou názvu kontejneru. Na základě názvu kontejneru lze konstruovat dotazy na konkrétní kontejner. [31]

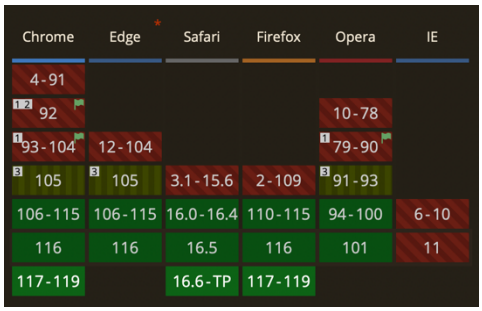
3.7.3 Výhoda Container Queries

Kontejnerové dotazy jsou užitečné zejména v případech responzivního designu, kdy jsou vytvářeny opakovaně používané komponenty. Například vytvoření komponenty karty produktu, která se může rozložit jedním způsobem, když je umístěna v postranním panelu a jiným způsobem, když je zobrazena na seznamu produktů. [32]

3.7.4 Podpora v prohlížečích

Container Queries nabízejí podporu ve všech moderních prohlížečích, výjimku představuje prohlížeč Internet Explorer od společnosti Microsoft, který funkcionalitu nepodporuje.

Obrázek 12 Podpora Container Queries



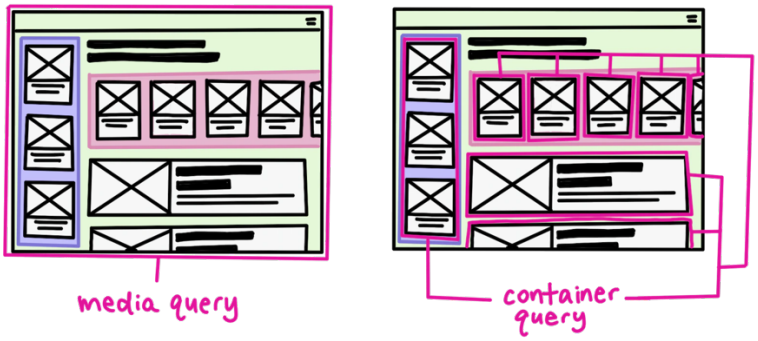
Zdroj: [20]

3.7.5 Porovnání s Media Queries

Narozdíl od Media Queries, které se vztahují k šířce obrazovky, Container Queries se vztahují k velikosti kontejneru, ve kterém se nachází obsah.

Pomocí mediálních dotazů je možné vytvořit responzivní prvek, který ve standardním případě vypadá dobře, avšak pravděpodobně nebude fungovat stejně dobře, pokud se přesune do kontejneru, který bude mít aplikovanou vlastnost CSS, která ovlivňuje rozměr. Pro správné fungování v takovém případě je potřeba přidat další CSS vlastnosti. Problém nově umožňují řešit Container Queries, pomocí dotazů na kontejner, lze cílit na jednotlivé prvky a umožnit jim přizpůsobit se jakémukoliv kontejneru. [33]

Obrázek 13 Rozdíl mezi Média a Container Queries



Zdroj: [32]

3.8 Knihovny pro tvorbu responzivního webu

CSS je jazyk stylů používaný k nastavování vzhledu HTML prvků na webové stránce. Knihovna kaskádních stylů je nástroj, který vývojářům umožní snadněji navrhovat webové stránky vyhovující standardům pomocí CSS. Narozdíl od vytváření projektu od úplného začátku, knihovna poskytuje nástroje, které umožnují rychlé vytvoření rozhraní.

3.8.1 Bootstrap

Jedná se o open-source projekt, který pomocí CSS a JavaScriptu poskytuje možnost vytváření responzivních webových stránek. Představuje jednu z nejdéle používaných knihoven, kterou v polovině roku 2010 vytvořil Mark Otto a Jackob Thorton ve společnosti Twitter. Cílem bylo vyvinout knihovnu, která umožní navrhovat responzivní weby, které by fungovaly na všech zařízeních. Bootstrap obsahuje řadu předpřipravených komponent včetně navigačních panelů, tlačítek, karuselů a modálů, které lze použít pouhým zkopírováním. To vývojářům pomáhá uspořit čas a úsilí, protože prvky nemusí vytvářet od úplného začátku. Místo toho mohou použít předpřipravené kusy kódů a přizpůsobit je svým potřebám. [34]

3.8.2 **Mřížka v Bootstrapu**

Mřížkový systém Bootstrapu používá k rozvržení a zarovnání řadu kontejnerů, sloupců a řádků. Je vytvořen pomocí flexboxu a je plně responzivní. Kontejnery umožňují horizontálně uspořádat obsah. Pro responzivní šířku v *px* je třída *container* nebo *container-fluid* pro plnou šířku na všech velikostech zobrazení. Řádky obalují sloupce a každý sloupec obsahuje padding pro kontrolu prostoru mezi nimi. Tento padding je na řádcích kompenzován zápornými marginy. Veškerý obsah ve sloupcích je vizuálně zarovnán doleva. V rozložení mřížky musí být obsah umístěn ve sloupcích a pouze sloupce mohou být potomkem řádků. Díky funkci flexboxu se sloupce bez zadané šířky automaticky roztáhnou na stejnou šířku. Šířky sloupců jsou nastaveny v procentech, takže jsou vždy pružné vůči svému nadřazenému prvku.

Pro zajištění responzivity mřížky existuje pět bodů zlomu:

- Extra small (do šířky 576px)
- Small (od šířky 576px)
- Medium (od šířky 768px)
- Large (od šířky 992px)
- Extra large (od šířky 1200px)

Body zlomu jsou založeny na mediálních dotazech s minimální šířkou, platí pro jeden bod zlomu a všech nad ním. Například pomocí třídy *col-sm-4* lze nastavit, že od velikosti obrazovky 576px bude sloupec zabírat 4/12 šířky řádku.
Tato část je zpracována podle [35].

3.8.3 **Komponenty v Bootstrapu**

Bootstrap obsahuje řadu předpřipravených komponent, které nabízí rychlý a jednoduchý vývoj responzivního layoutu. Pomocí JavaScriptové knihovny jQuery je možné komponenty ovládat. Například zavření vyskakovacího boxu pomocí *$().alert('close')*. Hlavní komponenty, které Bootstrap nabízí jsou:

Card

Jedná se o rozšiřitelný kontejner obsahu. Obsahuje možnosti pro záhlaví a zápatí a široké možnosti pro zobrazení. Nabízí například možnost definování podtitulku nebo akčních tlačítek ve spodní části.

Carousel

Nabízí posuvný modul pro procházení obsahu pomocí 3D CSS transformací. Pracuje s obrázky, textem nebo vlastními prvky. Obsahuje ovládací prvky pro kontrolu přechodů.

Modal

Představuje vyskakovací okno, ve kterém je možnost zobrazit jakýkoliv obsah. Disponuje rozhraním pro otevření/zavření. Obsahuje sekce rozdělené do tří částí (header, body, footer)

Tato část je zpracovaná podle [36].

3.8.4 Tailwind CSS

Tailwind CSS je moderní CSS knihovna, která umožňuje vývojářům vytvářet responzivní a elegantní webové stránky. Její unikátnost spočívá v přístupu k tvorbě rozhraní pomocí utilit tříd. Tailwind CSS je označován jako „utility-first framework“. [37]

Narozdíl od Bootstrapu, který obsahuje předdefinované komponenty, Tailwind CSS nabízí sadu jednoduchých utility tříd, které pokrývají vlastnosti CSS. Třídy jsou pojmenovány intuitivně, což vývojářům usnadňuje práci s nimi. Kombinací utility tříd lze dosáhnout komplexního designu bez nutnosti psaní vlastního CSS.

3.8.5 Historie Tailwind CSS

Nápad vytvořit Tailwind CSS vznikl z frustrace týmu z omezení tradičních CSS frameworků. Cílem bylo získat framework, který by poskytoval větší flexibilitu a kontrolu nad procesem návrhu, při zachování zjednodušeného pracovního postupu. První verze představila rozsáhlou sadu užitkových tříd, které pokrývaly širokou škálu aspektů, jako například typografie, barvy flexbox a další. Užitkové třídy byly navrženy tak, aby se lehce kombinovaly a umožňovaly tak vývojářům vytvářet vlastní návrhy bez nutnosti psaní vlastních kaskádních stylů. [38]

Jakmile si Tailwind získal svou popularitu, jeho komunita se rychle rozrostla. Vývojáři oceňovali detailní kontrolu nad použitými styly a možnost vytváření responzivního rozložení. Postupně se Tailwind vyvíjel a zdokonaloval, byly zavedeny nové funkce a další vylepšení, které reagovaly na zpětnou vazbu komunity. Začátkem roku 2021 představil Tailwind režim *Just-in-Time*, který zlepšil generování CSS na vyžádání a přinesl snížení velikosti výsledného souboru. [38]

Obliba frameworku vedla k integraci s různými nástroji a frameworky pro vývoj front-endu. Stal se oblíbenou technologií mezi vývojáři pracujícími s moderními frameworky JavaScriptu, jako jsou například React, Vue nebo Angular. [38]

3.8.6 Utility CSS

V utility first jde o psaní kódu pomocí jednoúčelových tříd. Většinou jde o třídy, které kombinují vlastnost CSS a její hodnotu. Například pro bílý text bychom měli třídu *text-white*. [39]

Na první pohled se může zdát, že se bude jednat o nepřehledný kus kódu, ve kterém je složité se orientovat, ale tento způsob přináší i řadu výhod, které například jsou:

- Absence nutnosti vymýšlení názvů CSS tříd
- Neroste velikost CSS souborů, jelikož se vlastnosti znovu používají mezi několika prvky, takže při přidání nového elementu se často využívá vlastností, které již kód obsahuje
- Bezpečnější provádění změn. Třídy v HTML narozdíl od globálních CSS jsou lokální, takže je možnost je změnit bez obav, že se změna projeví i na nežádoucích místech

Tato část je zpracována podle [40].

3.8.7 Konfigurace Tailwind CSS

Protože Tailwind CSS je knihovna pro vytváření rozhraní na míru, byl od začátku navrhnutý s ohledem na možnost přizpůsobení. Ve výchozím nastavení se v kořenovém adresáři projektu nachází soubor *tailwind.config.js*, který definuje vlastní úpravy. V rámci tohoto souboru lze například definovat vlastní barevná schémata nebo fonty písma. [41]

3.8.8 Výhody a nevýhody Tailwind CSS

Výhody

- Rychlejší proces vývoje
- Nízká velikost výsledného CSS souboru
- Snadno použitelný při znalosti kaskádních stylů
- Dobře zpracovaná dokumentace

Nevýhody

- Značení může u rozsáhlých projektů vypadat nepřehledně

- Vyžaduje znalost kaskádních stylů
- Nutnost vytvářet vše od začátku

Tato část je zpracovaná podle [42].

3.9 Vykreslování webové stránky

Rychlé webové stránky poskytují lepší uživatelský zážitek. Uživatelé chtějí a očekávají webové rozhraní, které se rychle načítá a plynule interaguje. Hlavní problémy v oblasti výkonu webových stránek souvisí s latencí. [43]

Prohlížeče jsou většinou považovány za jednovláknové. To znamená, že vykonají úlohu od začátku do konce, než se převezme další úloha. Klíčovým prvkem je doba vykreslování, která zajišťuje, že hlavní vlákno dokáže dokončit veškerou zadanou práci a přitom být stále k dispozici pro zpracování interakcí s uživatelem. Zlepšení výkonu lze dosáhnout minimalizací zátěže hlavního vlákna, aby se zajistilo plynulé vykreslení a okamžité reakce na uživatelské akce. [43]

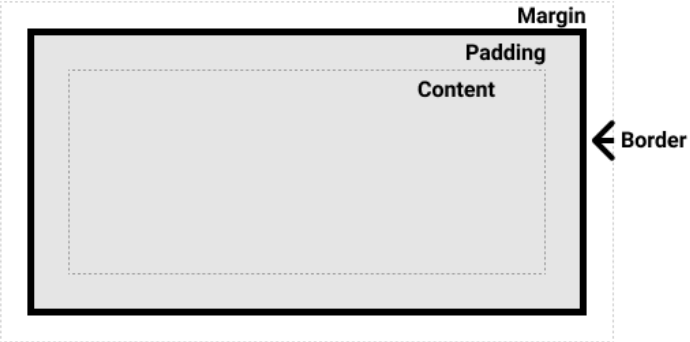
3.9.1 CSS Box model

Každý HTML element v dokumentu představuje obdélník neboli box. Ten je tvořen obsahem, který je označován jako *content*. Obsah je obalen třemi oblastmi, které mají dané pořadí – *padding*, *border* a margin. Tyto oblasti jsou součástí každého prvku na webové stránce. Pokud nejsou v CSS definovány, obsahují nulové hodnoty. Jednotlivým oblastem lze nastavit rozměr, typ pozicování nebo vztah k ostatním elementům.

- **Content** – Obsahuje vlastní obsah prvku. Výšku a šířku udává velikost obsahu. Obsahem může být například fotka, text nebo video.
- **Padding** – Vyhrazuje prostor okolo obsahu, až po ohraničení obsahu. Jedná se o odsazení prvku.
- **Border** – Představuje samotné ohraničení elementu. Nachází se mezi vnitřním a vnějším odsazením. Používá se k dekoraci a je možné ohraničení vizuálně upravovat.
- **Margin** – Vnější okraj, který rozšiřuje prostor okolo elementu. Používá se k oddělení jednotlivých prvků.

Tato část je zpracována podle [44].

Obrázek 14 CSS Box model



Zdroj: [45]

3.9.2 Vykreslovací jádro

Jádrem každého prohlížeče je vykreslovací jádro, které bývá označováno jako rozvrhovací jádro nebo jádro prohlížeče. Vykreslovací jádro je důležité z důvodu interpretace a vykreslování webového obsahu, kdy transformuje HTML, CSS a JavaScript do vizuálně přívětivých a interaktivních webových stránek. [46]

Vykreslovací jádro je zodpovědné za zpracování strukturálních a prezentačních aspektů webových stránek. Určuje, jakým způsobem se zobrazí prvky na obrazovce a jak se zpracovávají interakce od uživatele. Funguje jako spojení mezi prohlížečem a obsahem webové stránky a usnadňuje převod značkovacích jazyků do vizuální reprezentace. Jednotlivé prohlížeče používají různá vykreslovací jádra, každé s unikátní implementací a odlišnou optimalizační strategií. [46]

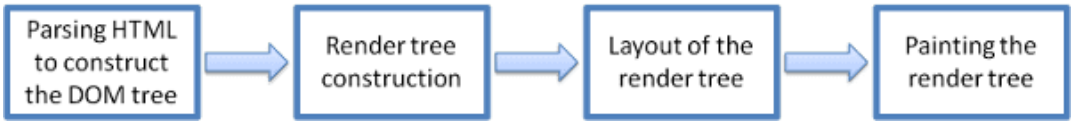
Mezi obvyklé vykreslovací jádra se řadí:

- **Webkit** – vyvinutý společností Apple, je základem prohlížečů jako je Safari, původně odvozen z jádra KHTML
- **Blink** – vyvinutý společností Google, vychází z Webkitu, představuje vykreslovací jádro pro prohlížeč Google Chrome a Microsoft Edge
- **Gecko** – vyvinutý společností Mozilla, je základem pro prohlížeč Firefox
- **Trident** – dříve využívaný v prohlížeči Internet Explorer, byl nahrazen systémem Blink v prohlížeči Microsoft Edge
- **EdgeHTML** – vlastní vykreslovací jádro společnosti Microsoft používané ve starších verzích prohlížeče Microsoft Edge před přechodem na Blink

3.9.3 Proces vykreslení

Načtení webové stránky začíná požadavkem na server, který vrátí HTML. Prohlížeč začne analyzovat a převádět přijaté bajty do stromu DOM. Pokud při zpracování narazí na externí zdroje, například soubory stylů, skripty nebo odkazy na vložené obrázky, zašle další požadavek na získání potřebných dat. Některé požadavky mohou být blokující. V tom případě je zpracování HTML pozastaveno, dokud není zdroj zcela načtený. Jakmile je vytvořen DOM, prohlížeč pokračuje konstruováním objektového modelu CSS. [47]

Obrázek 15 Proces vykreslení



Zdroj: [48]

Informace o stylu spolu s vizuálními instrukcemi v HTML budou použity k dalšímu kroku, vytvoření vykreslovacího stromu. Vykreslovací strom obsahuje obdélníky s vizuálními atributy jako například barva a rozměry, které jsou ve správném pořadí vykresleny na obrazovce. Po sestavení vykreslovacího stromu nastává proces rozvržení. Každému uzlu se zadají přesné souřadnice, kde se má na stránce zobrazit. Následuje fáze malování, kde se projde vykreslovací strom a každý uzel se vykreslí pomocí vrstvy uživatelského rozhraní. Pro lepší uživatelský komfort vykresluje jádro obsah co nejdříve. Nečeká se na analýzu celého HTML dokumentu. Jednotlivé části obsahu jsou analyzovány a zobrazeny, zatímco proces pokračuje zbytkem obsahu, který přichází ze síťové vrstvy. [48]

3.9.4 Zpracování

Konstrukce stromu DOM

Konstrukce DOM je postupná. Zdrojový kód HTML se mění na uzly, které se mění na strom DOM. Každý uzel začíná tokenem počáteční značky a je zakončen tokenem koncové značky. Jednotlivé uzly obsahují všechny důležité informace o HTML prvku, které jsou popsány pomocí tokenů. Uzly jsou propojeny na základě hierarchie tokenů. Strom DOM popisuje obsah dokumentu. Prvním a zároveň kořenovým uzlem stromu je HTML element <*html*>. Strom reflektuje hierarchii mezi jednotlivými prvky. Uzly vnořené do jiných uzlů jsou podřízené uzly. Čím větší je počet uzlů DOM, tím déle trvá sestavení výsledného stromu. [49]

Konstrukce stromu CCSOM

Druhým krokem při vykreslování je zpracování CSS a sestavení a zpracování CSSOM stromu. Objektový model CSS je podobný modelu DOM. Jedná se o nezávislé datové struktury. Prohlížeč transformuje CSS pravidla do mapy stylů, které rozumí a je schopen s ní pracovat. Prochází jednotlivá pravidla a generuje strom uzlů s rodičovskými, podřízenými a sourozeneckými vztahy na základě CSS selektorů. Webový prohlížeč začíná nejobecnějším pravidlem použitelným pro uzel a postupně rekurzivně zpřesňuje vypočtené styly použitím konkrétních pravidel. [49]

Kompilace JavaScriptu

Mezitím, kdy probíhá tvorba stromu DOM a CCSOM se stahují další prostředky včetně JavaScriptu, který se parsuje, kompiluje a interpretuje. Jednotlivé skripty jsou parsovány do syntaktických stromů. Některé jádra prohlížečů předají syntaktický strom kompilátoru, který generuje bajtový kód. Tento proces se označuje jako kompilace JavaScriptu. Většina kódu je zpracována v hlavním vlákně, avšak existují výjimky, kdy je například kód spuštění v takzvaných Web Workers. [49]

3.9.5 Vykreslení

Část vykreslení se skládá z kroků styly, rozvržení, malování a v některých případech i kompozice. Stromu DOM a CSSOM vytvořené v přechozím kroku se spojí do vykreslovacího stromu, který se následně použije k výpočtu každého viditelného prvku. [49]

Styly

Elementy, jako například <*head*> včetně potomků nebo s vlastností *display* nastavenou na hodnotou *none* se ve stromu vykreslení nezahrnují, protože se ve výsledném zobrazení nenacházejí. Elementy s vlastností *visibility* a hodnotou *hidden* jsou součástí vykreslovacího stromu, jelikož zabírají dostupné místo. Pro každý viditelný element jsou aplikována pravidla ze stromu CCSOM. [49]

Rozvržení

Dalším krokem je spuštění rozvržení vykreslovacího stromu za účelem výpočtu geometrie každého elementu. Rozvržení představuje proces, kdy se u elementů vypočítají rozměry a umístění všech elementů ve vykreslovacím stromě. [50]

Na webové stránce je většina prvků ve formě boxů. Různá zařízení a různé nastavení počítačů představují neomezený počet kombinací velikostí zobrazení. V této fázi se na základě velikosti viewportu definuje, jaké budou rozměry boxů. [49]

Malování

Po výpočtu rozvržení se přejde ke kroku malování. V této fázi se převede každý prvek vypočtený ve fázi rozvržení na skutečné pixely na obrazovce. Malování představuje vykreslení každé vizuální části na obrazovku, včetně barev, textu, stínů nebo prvků jako jsou tlačítka a obrázky. [49][50]

Kompozice

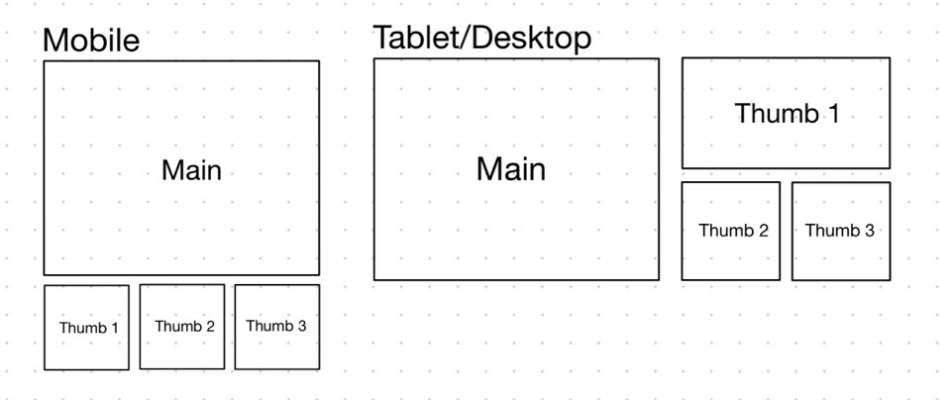
V případě, kdy jsou jednotlivé části dokumentu vykresleny v různých vrstvách, které se navzájem překrývají, je nutné použít kompozici. Tím se zajistí jejich vykreslení na obrazovku ve správném pořadí. [49]

4 Vlastní práce

4.1 Responzivní galerie

Responzivní galerie představuje webový prvek, který se používá k zobrazování fotografií, videí či jiného vizuálního obsahu v rámci webových stránek. Hlavním úkolem je přizpůsobit své zobrazení různým velikostem obrazovek. V příkladu bude vytvořena galerie, která obsahuje hlavní obrázek a tři náhledové obrázky.

Obrázek 16 Návrh galerie

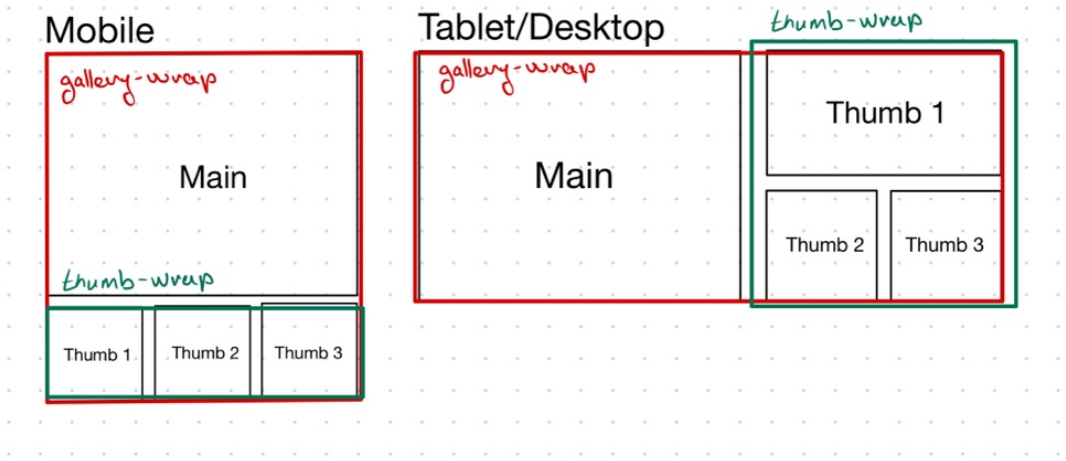


Zdroj: vlastní

4.1.1 Implementace pomocí modulu flexbox

Pro implementaci pomocí flexboxu je potřeba na začátku zvolit vhodnou strukturu HTML značek.

Obrázek 17 Návrh implementace galerie pomocí flexboxu



Zdroj: vlastní

Nejprve bude obalen hlavní obrázek se skupinou náhledů do HTML značky *div*, na které bude nastavena třída *gallery-wrap*. Dále bude vytvořena skupina jednotlivých náhledů pomocí značky *div* a nastavena třída *thumb-wrap*. Na hlavní obrázek a náhledové fotky bude aplikována specifická třída, která následně bude sloužit k aplikování stylů.

Zdrojový kód 1 HTML pro galerii pomocí flexboxu

```
<div class="gallery-wrap">
    <img src="./image.jpg" alt="Main image" class="main-image">
    <div class="thumb-wrap">
      <img src="./image.jpg" alt="Main image" class="thumb-image thumb-1">
      <img src="./image.jpg" alt="Main image" class="thumb-image thumb-2">
      <img src="./image.jpg" alt="Main image" class="thumb-image thumb-3">
    </div>
  </div>
```

Zdroj: vlastní

Aby se obalové položky chovaly jako flex kontejnery bude jim nastavena vlastnost *display* na hodnotu *flex*. Jako bod zlomu bude zvolena hodnota 640 px, při které se aplikuje adaptivní vzhled. Pro dosažení zobrazení položek vedle sebe bude změněna vlastnost *flex-direction* na hodnotu *row.* Je důležité poznamenat, že v příkladech při implementaci CSS je využitý přístup **mobile-first**, který se vyznačuje psaním CSS nejprve pro mobilní zařízení a následně pro větší rozměry obrazovek.

Zdrojový kód 2 Ukázka z implementace pomocí flexboxu

```
.m-wrap {
  display: flex;
  flex-direction: column;
  gap: 10px;
}

.thumb-wrap {
  display: flex;
  gap: 10px;
  width: 100%;
}
```

```
@media (min-width: 640px) {
  .gallery-wrap {
    flex-direction: row;
    height: 300px;
  }
  .thumb-wrap {
    flex-wrap: wrap;
  }
}
```

Zdroj: vlastní

Pro vytvoření náhledových fotek na mobilních zařízeních nebylo nutno nastavit vlastnost *flex-direction*, jelikož ve výchozím stavu má hodnotu *row*, kterou v příkladu potřebujeme. Na větších zařízeních bude u kontejneru náhledových fotek vlastnost *flex-wrap* na hodnotu *wrap*, čímž dojde k zalomení obrázků, které se na daný řádek nevejdou. Pro vytvoření efektu prvního náhledového obrázku samostatně přes celou šíři byla

nastavena vlastnost *width* na hodnotu *100 %*. Zbylé náhledové fotky mají nastavenou poloviční šířku 50 %.

Obrázek 18 Výsledek galerie na mobilu

Zdroj: vlastní

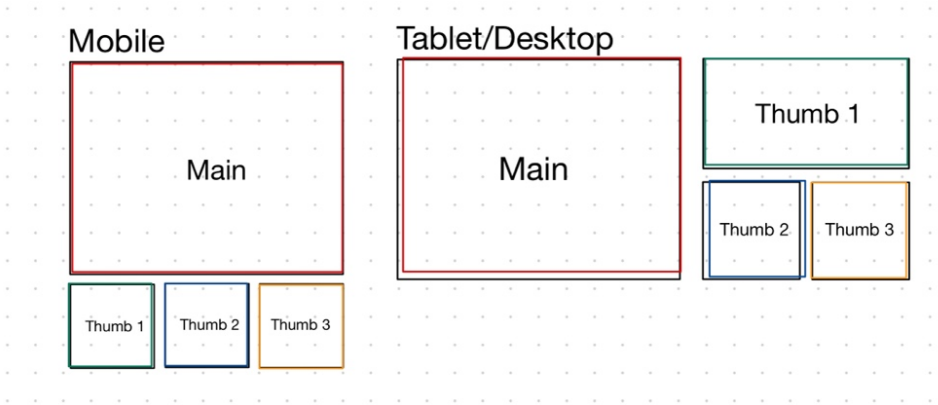
Obrázek 19 Výsledek galerie na větších zařízeních

Zdroj: vlastní

4.1.2 **Implementace pomocí modulu grid**

Opět bude vytvořena mřížka, akorát tentokrát pomocí CSS modulu, který byl navržen pro usnadnění tvorby layoutu. Obdobně jako u předchozího příkladu bude na začátku navrhnuto řešení. Modul grid nabízí možnost definovaní oblastí, které v implementaci budou využity. První bude *main*, která představuje hlavní obrázek a následně pro každý náhledový obrázek bude definována oblast.

Obrázek 20 Návrh implementace galerie pomocí gridu



Zdroj: vlastní

Stejně jako u flexboxu nejprve bude nastavena vlastnost *display*. Dalším krokem je definovat, kde budou prvky umístěny, k tomu bude využita vlastnost *grid-template-areas*. Za účelem vytvoření responzivního designu bude definován bod zlomu. V ten okamžik bude upraveno zobrazení mřížky dle návrhu.

Zdrojový kód 3 Ukázka definice vlastnosti grid-template-areas

```
.image-wrap {
  display: grid;
  grid-template-areas: "main-image main-image main-image"
                       "thumb-image-1 thumb-image-2 thumb-image-3";
}
@media (min-width: 640px) {
  .image-wrap {
    grid-template-areas: "main-image thumb-image-1 thumb-image-1"
                         "main-image thumb-image-2 thumb-image-3";
  }
}
```

Zdroj: vlastní

Po nastavení vlastnosti *grid* se každý prvek stane součástí mřížky. Na základě předem vytvořených oblastí bude přiřazen potomek k určité oblasti. Pro přiřazení bude využita vlastnost *grid-area* s hodnotou odpovídající dané oblasti. Tím dojde k dosažení prvotního rozdělení prvků do mřížky, které je snadno rozeznatelné na první pohled v CSS kódu.

Zdrojový kód 4 Ukázka přiřazení potomků do oblastí

```
.main-image { grid-area: main-image; }
.thumb-image-1 { grid-area: thumb-image-1; }
.thumb-image-2 { grid-area: thumb-image-2; }
.thumb-image-3 { grid-area: thumb-image-3; }
```

Zdroj: vlastní

V dalším kroku budou nastaveny rozměry obrázků. Zde budou využity vlastnosti *grid-template-rows* a *grid-template-rows*, které určují velikosti mřížky pro řádky a sloupce.

Zdrojový kód 5 Definice velikosti řádků a sloupců

```
.image-wrap {
  grid-template-rows: 300px 100px;
}
@media (min-width: 640px) {
  .image-wrap {
    grid-template-columns: 300px minmax(150px, 1fr) minmax(150px, 1fr);
    grid-template-rows: 145px 145px;
  }
}
```

Zdroj: vlastní

Na větších obrazovkách pro definici velikosti sloupců byla využita CSS funkce *minmax()*, která poskytuje možnost definovaní minimální a maximální velikosti. Jelikož galerie se na větších zařízeních zobrazuje přes celou stránku. Byla využita jednotka *fr*, která využívá veškerý zbylý prostor.

Řešení je vizuálně totožné s implementací pomocí flexboxu.

4.1.3 Srovnání implementací

Tabulka 1 Srovnání implementací responzivní galerie

Hodnocené kritérium	**Flexbox**	**Grid**
Struktura HTML	Prvky galerie jsou ve více úrovních	Prvky galerie se nacházejí na stejné úrovni
Čitelnost kódu	Delší HTML i CSS kód, ztráta přehlednosti	Definice mřížky pomocí vlastnosti *grid-template-areas* nabízí čitelný CSS kód
Nastavení velikosti obrázků	Pro výpočet velikostí obrázku použité CSS funkce	Velikost obrázků je řízena nastavením velikosti mřížky (vlastnosti *grid-template-columns* a *grid-template-rows*)

Zdroj: vlastní

Implementace pomocí modulu grid má výhodu jednoduššího zápisu sémantických značek. Narozdíl od modulu flexbox nebylo potřeba zanořovat HTML elementy.

V implementaci gridové mřížky bylo možné využít vlastnost *grid-template-areas*, která z pohledu čitelnosti kódu je přehlednější než zápis pomocí flexboxu. Na první pohled z definice mřížky lze poznat, kde se prvek nachází.

V případě definice šířky obrázků bylo možné u modulu grid využít k definici šířky vlastnosti pro nastavení šířky sloupců a výšky řádků a u obrázků pouze nastavit, aby zabíraly 100 % šířky a výšky. U řešení pomocí flexboxu byla šířka a výška nastavena na jednotlivých elementech. V případě budoucí úpravy by řešení pomocí flexboxu vyžadovalo vyšší náročnost.

V tomto případě pro tvorbu responzivní galerie je vhodnější využít modul grid, který nabízí přehlednější zápis, nevyžaduje složitou strukturu HTML elementů a nabízí jednoduchý zápis velikosti obrázků, který může být snadno upravitelný.

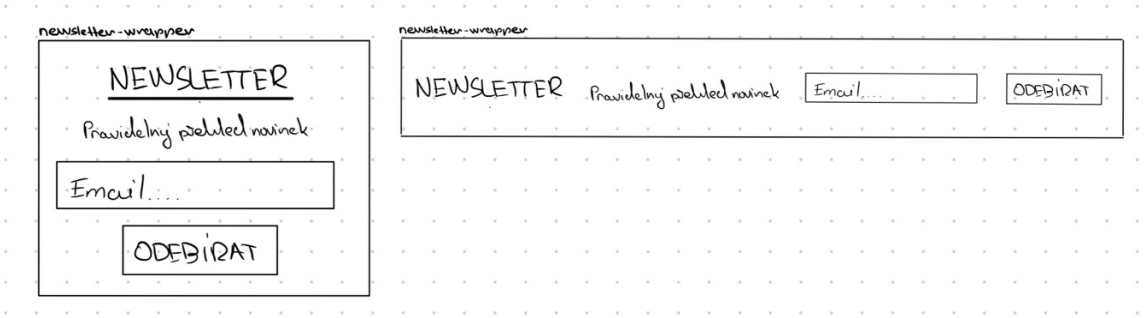
4.2 Tvorba responzivní komponenty pomocí Container Queries

V následujícím příkladu bude navržena a implementována tradiční komponenta elektronického obchodu na získávání emailů od uživatelů za účelem zasílání marketingových sdělení.

Tvorba komponenty pomocí techniky Container Queries představuje flexibilní přístup k vytváření responzivního designu. Předpoklad v příkladu je, že předem není známo, kde bude na stránce komponenta umístěna a kolik bude mít dostupného prostoru. Container Queries umožňují vytvářet komponenty, které přizpůsobí vzhledu na základě šířky svého rodičovského prvku.

Nejprve bude navržen vzhled komponenty. Ta bude obsahovat nadpis, popis a pole pro zadání emailu s tlačítkem pro odeslání. Opět bude navržena pomocí přístupu *mobile-first*.

Obrázek 21 Návrh komponenty pro newsletter



Zdroj: vlastní

Zdrojový kód HTML se bude skládat z hlavního kontejneru, na kterém bude aplikována vlastnost *container* z Container Quries. Dále bude obsahovat blok, který bude mít unitř dvě textové značky pro nadpis a popis a formulář pro odeslání emailu.

Zdrojový kód 6 HTML pro newsletter komponentu

```
<div class="newsletter-container">
    <div class="newsletter">
      <span class="title">Newsletter</span>
      <span class="description">Pravidelný přehled novinek</span>
      <form action=".">
        <label>
          <input type="email" name="email" placeholder="Email...">
        </label>
        <button type="submit">Odebírat</button>
      </form>
    </div>
</div>
```

Zdroj: vlastní

Rodičovskému prvku se třídou *newsletter-container* nejprve bude nastaven kontejner pro šířku *inline-size* a bude pojmenován pomocí vlastnosti *container-name* na hodnotu newsletter. Tento název bude klíčový při použití dotazu na šířku kontejneru.

Zdrojový kód 7 Vytvoření a pojmenování kontejneru

```
.newsletter-container {
  container-type: inline-size;
  container-name: newsletter;
}
```

Zdroj: vlastní

Alternativně lze tento zápis zkrátit na jeden řádek pomocí vlastnosti *container*, které je nejprve nastaven název kontejneru a následně znak / a zvolen typ kontejneru.

Zdrojový kód 8 Alternativní vytvoření a pojmenování kontejneru

```
.newsletter-container {
  container: newsletter / inline-size;
}
```

Zdroj: vlastní

Dále bude definováno chování rozložení prvků. Při menším zobrazení budou položky vyskládány pod sebe. V tomto příkladu bude využito modulu flexbox, protože z návrhu je patrné, že se jedná jednodimenzionální zobrazení. Nejprve bude využita vlastnost *display* s hodnotou *flex* kombinována s vlastností *flex-direction* a hodnotou *column*. Pro získání mezery mezi prvky bude aplikována vlast *gap* s hodnotou *20px*.

Zdrojový kód 9 Nastavení rozložení prvků

```
.newsletter-container .newsletter {
  display: flex;
  flex-direction: column;
  gap: 20px;
}
```

Zdroj: vlastní

Pro vytvoření responzivního designu bude využito dotazu na šířku kontejneru a přeskupení položek do horizontální linie.

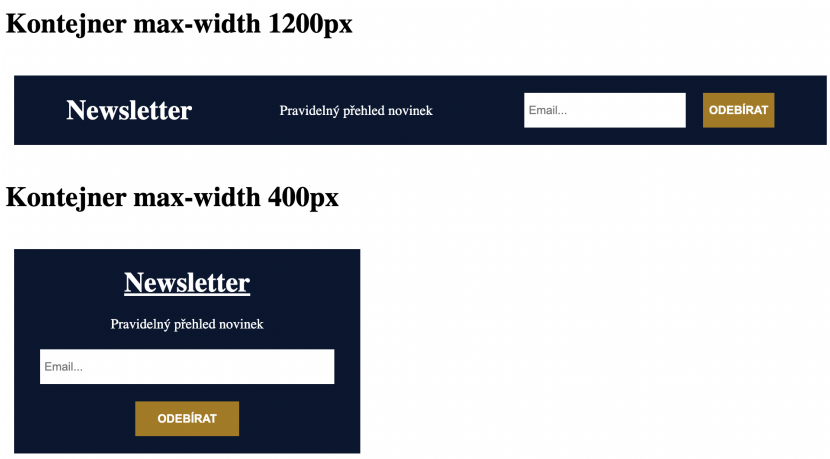
Zdrojový kód 10 Dotaz na šířku kontejneru

```
@container newsletter (min-width: 830px) {
  .newsletter-container .newsletter {
    justify-content: space-between;
    align-items: center;
    flex-direction: row;
  }
}
```

Zdroj: vlastní

Pokud bude šířka kontejneru alespoň 830 px, aplikují se pravidla uvnitř. Rozvržení bude převedeno do řádkové formy a položky budou vycentrovány podle hlavní osy. Mezera mezi položky bude nastavena na hodnotu *space-between*, tím dojde k využití volného místa a roztažením položek podél celého řádku.

Obrázek 22 Výsledek newsletter komponenty



Zdroj: vlastní

Na výsledném obrázku je vidět, jak se komponenta přizpůsobuje, pokud je umístěna v rodičovském prvku, který má šířku 1200 px a 400 px. Takto byla vytvořena komponenta, která může být umístěna na jakémkoliv místě webové stránky.

4.2.1 SWOT analýza použití techniky Container Queries

Tabulka 2 SWOT analýza použití techniky Container Queries

Silné stránky (Strengths)	**Slabé stránky (Weaknesses)**
- Flexibilita - Udržovatelnost - Možnost umístění na libovolné místo	- Kompatibilita - Implementace - Riziko chyby
Příležitosti (Opportunities)	**Hrozby (Threats)**
- Modularita - Mobilní aplikace	- Nedostatek podpory - Příchod nové technologie

Zdroj: vlastní

Silné stránky

Flexibilita – Container Queries umožňují definovat bod zlomu na základě velikosti rodičovského prvku (kontejneru)

Udržovatelnost – možnost definovat chování na úrovni komponent může zlepšit udržovatelnost zdrojového kódu

Možnost umístění na libovolné místo – Container Queries nabízejí možnost umístění komponenty na libovolné místo, jelikož vzhled je založen na dostupném prostoru

Slabé stránky

Kompatibilita – Container Queries nepodporují straší verze webových prohlížečů

Implementace – Implementace vyžaduje vyšší časové nároky

Riziko chyb – při implementaci nových technologií může dojít k chybám a nekonzistentnosti

Příležitosti

Modularita – použití Container Queries může přispět k vyšší modularitě tvorby komponent, které lepší možností pro kombinaci

Mobilní aplikace – usnadnění vývoje PWA mobilních aplikací

Hrozby

Nedostatek podpory – pokud se Container Queries neuchytí jako standard, může jejich podpora klesnout

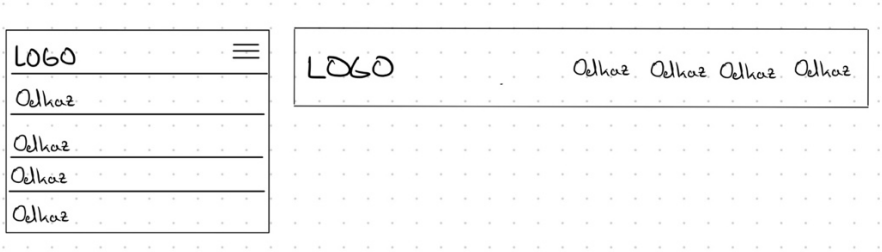
Příchod nové technologie – možnost nahrazení současného trendu novou technologií

4.3 Navigační menu

Navigační menu představuje základní prvek většiny webových stránek. Jedná se o rozhraní, které umožňuje uživateli se pohybovat po online prostředí a nalézat hledaný obsah.

Navigace se bude skládat z loga a skupiny odkazů. Logo bude umístěno v levém horním rohu. Na menších zařízeních se budou odkazy skládat pod sebe a zabírat celou šířku obrazovky, zatímco na větších zařízeních budou položky vyskládány vedle sebe.

Obrázek 23 Návrh navigačního menu



Zdroj: vlastní

Pro zvýšení přístupnosti webu bude využita HTML značka *nav*, ta uživatelům se zrakovými či jinými omezenými schopnosti pomůže se snadno navigovat v rámci navigačního obsahu. Značka současně zvyšuje kvalitu SEO na našem webu, jelikož vyhledávače, jako například Google, často preferují webové stránky se správně zvolenou sémantikou. K vytvoření ikonky navigace na menších zařízeních, bude využito HTML a CSS.

Zdrojový kód 11 HTML pro navigační menu

```
<nav class="navigation">
  <div class="logo-wrapper">
    <a href="#" class="logo">LOGO</a>
    <div class="hamburger-icon">
      <div class="line"></div>
      <div class="line"></div>
      <div class="line"></div>
    </div>
  </div>
  <div class="links">
    <a href="#">Odkaz 1</a>
    <a href="#">Odkaz 2</a>
    <a href="#">Odkaz 3</a>
    <a href="#">Odkaz 4</a>
  </div>
</nav>
```

Zdroj: vlastní

K implementaci CSS bude zvolen modul flexbox, jelikož se jedná o jednoduchý layout, který bude buď ve formě řádku nebo sloupce.

Nejprve na třídě *navigation* bude nastavena vlastnost *display* na hodnotu *flex*. Následně bude aplikována vlastnost *flex-direction* na hodnotu *column*, pro vytvoření sloupcového zobrazení. Opět, jako v předchozích příkladech, bude využita technika **mobile-first**. Na větší zařízeních bude flex kontejner orientován do řádku. Současně na něm bude nastavena vlastnost *justify-content* na hodnotu *space-between*. Tím vznikne mezera mezi logem a položkami navigačního menu.

Zdrojový kód 12 Nastavení vlastnosti flex na třídu navigation

```
.navigation {
  display: flex;
  flex-direction: column;
}
```

```
@media (min-width: 600px) {
  .navigation {
    align-items: center;
    flex-direction: row;
    justify-content: space-between;
  }
}
```

Zdroj: vlastní

K umístění loga a ikonky bude využito opět modulu flexbox, jelikož se jedná o řádek a cílem je logo a ikonku oddělit mezerou. Zde se bude hodit nastavení vlastnosti *justify-content* a nastavení vlastnosti *align-items*, pro vycentrování podél hlavní osy. Jelikož ikona bude pouze na mobilních zobrazeních, bude pomocí mediálního dotazu nastavena na třídě *hamburger-icon* vlastnost *display* na hodnotu *none*.

Zdrojový kód 13 Nastavení loga a ikonky

```
.logo-wrapper {
  align-items: center;
  display: flex;
  justify-content: space-between;
}
```

```
@media (min-width: 600px) {
  .hamburger-icon {
    display: none;
  }
}
```

Zdroj: vlastní

Jednotlivé odkazy budou na menších obrazovkách ve sloupcovém layoutu a na větších obrazovkách v řádkovém. Opět ideální scénář pro modul flexbox. Na element se třídou links bude pomocí JavaScriptu aplikována třída *active* pokud uživatel klikne na ikonku. Ve chvíli, kdy bude mít element obě třídy bude na menších obrazovkách zobrazeno menu. U větších obrazovek bude vlastnost *display* nastavena na hodnotu *flex* vždy.

Zdrojový kód 14 Nastavení CSS vlastností na odkazy v menu

```
.links {
  display: none;
  flex-direction: column;
}

.links.active {
  display: flex;
}
```

```
@media (min-width: 600px) {
  .links {
    display: flex;
    flex-direction: row;
  }
}
```

Zdroj: vlastní

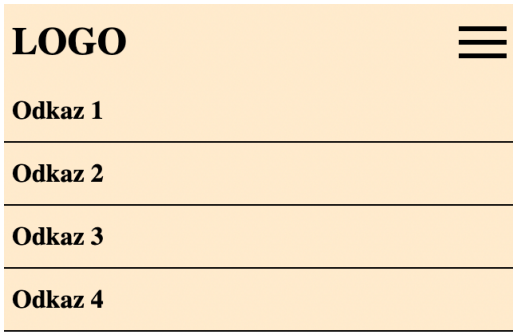
Obrázek 24 Výsledné zobrazení menu na větších zařízeních

Zdroj: vlastní

Obrázek 25 Výsledné zobrazení menu na mobilních zařízeních – zavřené

Zdroj: vlastní

Obrázek 26 Výsledné zobrazení menu na mobilních zařízeních – otevřené



Zdroj: vlastní

4.3.1 Zhodnocení implementace

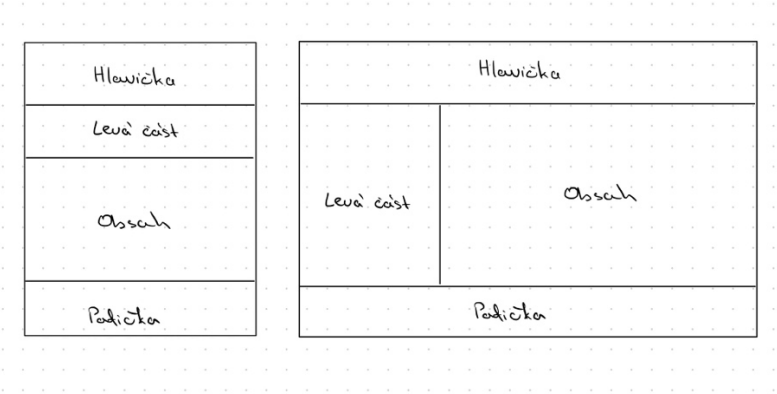
V příkladu bylo implementováno jednoduchá navigační menu pomocí modulu flexbox. Výsledek splňuje podmínky responzivního návrhu. Flexbox poskytuje dostatečné možnosti pro tvorbu základního navigačního rozhraní.

Jednou z hlavních výhod modulu flexbox je jeho flexibilita. Umožňuje snadnou manipulaci s pořadím a zarovnáním prvků. Díky tomu je snadné přizpůsobit navigační menu různým zařízením.

4.4 Rozvržení s levým bočním panelem

V následujícím příkladu bude implementováno rozvržení webové stránky s jedním bočním panelem umístěným na levé straně. Levý sloupec se obvykle navigační nabídku nebo v případě internetových obchodů možnost pro filtrování produktů. Vedle levého panelu se bude nacházet místo pro obsah webové stránky. V rámci rozvržení bude také implementována hlavička a patička webu.

Obrázek 27 Návrh rozvržení s levým bočním panelem



Zdroj: vlastní

Stejně jako v přechozích příkladech bude využito přístupu mobile-first.

4.4.1 Implementace pomocí modulu flexbox

Zdrojový HTML kód bude pro hlavičku obsahovat značku header, levý panel bude mít značku *section*. Obsah bude umístěn uvnitř značky *main* a patička webu bude ve značce *footer*. Pro implementaci pomocí flexboxu bude na značce *body*, která bude obalovat zmíněné značky nastavena třída *left-flexbox-layout*.

Zdrojový kód 15 HTML pro rozložení s levým sloupcem – flexbox

```
<body class="left-flexbox-layout">
  <header class="header">Hlavička</header>
  <section class="left-section">Levá část</section>
  <main class="content">Obsah</main>
  <footer class="footer">Patička</footer>
</body>
```

Zdroj: vlastní

Pro aktivaci modulu flexbox bude nastavena na třídě *left-flexbox-layout* vlastnost *display* na hodnotu *flex*. Následně bude definováno rozložení do sloupce pomocí vlastnosti *flex-direction* a hodnoty *column*. Pro ukázku bude na třídě také nastavena vlastnost *min-height* na hodnotu *100 vh*. Jednotka *vh* představuje relativní procentuální výšku obrazovky a její hodnota představuje procento z výšky obrazu. Tím bude dosaženo roztažení layoutu přes celou stránku.

Zdrojový kód 16 Nastavení vlastnosti flex

```
.left-flexbox-layout {
  display: flex;
  flex-direction: column;
  min-height: 100vh;
}
```

Zdroj: vlastní

U jednotlivých položek bude definována výška a barva pro rozlišení v rámci ukázky. U elementu *main* se třídou *content* bude výška nastavena pomocí vlastnosti *flex* na hodnotu *1*. Tím bude zabírat maximální možnou zbylou výšku.

Zdrojový kód 17 Nastavení velikostí položek

```
.header {
  background: #EC8F5E;
  height: 50px;
}

.footer {
  background: #9FBB73;
  height: 50px;
}
```

```
.left-section {
  background: #F1EB90;
  height: 100px;
}

.content {
  background: #F3B664;
  flex: 1;
}
```

Zdroj: vlastní

Tím bude dosaženo požadovaného rozložení na menších obrazovkách.

Pro zobrazení v rámci na větších obrazovkách bude nastavena vlastnost *flex-direction* na hodnotu *row* a *flex-wrap* na hodnotu *wrap*. Tím bude dosaženo zalomení přetékajících elementů.

Zdrojový kód 18 Nastavení zalomení položek

```
@media (min-width: 750px) {
  .left-flexbox-layout {
    flex-direction: row;
    flex-wrap: wrap;
  }
}
```

Zdroj: vlastní

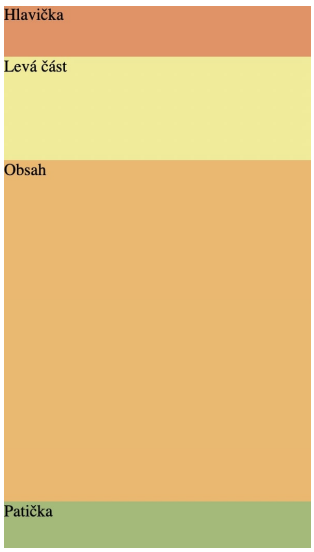
U elementu se třídou *header* a *footer* bude nastavena výška 80 px a šířka 100 %. Tím bude prvek zabírat celou šířku obrazovky. U postranního panelu bude šířka nastavena na 200 px a u hlavního obsahu k definici šířky bude využita CSS funkce *calc*. Ta bude obsahovat rozdíl 100 % šířky a 200 px, které zabírá levý panel. Následně u elementů se třídou *content* a *left-section* bude nastatvena výška opět pomocí funkce *calc* s rozdílem 100 % a 160 px. Hodnota 160 px představuje součet výšky elementů se třídou *header* a *footer*.

Zdrojový kód 19 Nastavení velikostí položek flexboxu na větších zařízeních

```
@media (min-width: 750px) {
.left-flexbox-layout .left-section, .left-flexbox-layout .content {
    height: calc(100vh - 160px);
  }

  .left-flexbox-layout .header, .left-flexbox-layout .footer {
    height: 80px;
    width: 100%;
  }

  .left-flexbox-layout .left-section {
    width: 200px;
  }

  .left-flexbox-layout .content {
    width: calc(100% - 200px);
  }
}
```

Zdroj: vlastní

Obrázek 28 Výsledné zobrazení rozložení s levým panelem na mobilu



Zdroj: vlastní

Obrázek 29 Výsledné zobrazení rozložení s levým panelem na desktopu

Zdroj: vlastní

4.4.2 Implementace pomocí modulu grid

Zdrojový kód HTML bude obdobný, jako při implementaci pomocí flexboxu. Bude upraven pouze název třídy na elementu *body* na hodnotu *left-grid-layout*.

Pro zapnutí modulu bude nastavena vlastnost display na hodnotu *grid*. Dále bude definována mřížka pomocí vlastnosti *grid-template-areas*. Na mobilních zařízeních bude mřížka obsahovat pouze jeden sloupec, zatímco na větších zařízeních bude mřížka rozdělena do sloupců dvou. Pro ukázku bude na mobilních zařízeních výška hlavičky a patičky nastavena na 50 px, výška levého panelu na 100 px a sekce pro obsah bude vyplňovat zbylý prostor stránky. K nastavení těchto hodnot bude využito vlastnosti *grid-template-rows*, která definuje velikosti řádků v mřížce. Na větších zařízeních bude levý sloupec mít šířku 200 px a hlavička a patička výšku 80 px. Výška obsahu a levé části bude zabírat zbylý prostor stránky. Bude upravena vlastnost *grid-template-rows* a přidána vlastnost *grid-template-columns* pro definování šířky sloupců.

Zdrojový kód 20 Definice mřížky pro rozložení s levým bočním panelem

```
.left-grid-layout {
  display: grid;
  grid-template-rows: 50px 100px 1fr 50px;
  grid-template-areas: "header"
                       "left-section"
                       "content"
                       "footer";
}
@media (min-width: 750px) {
  .left-grid-layout {
    grid-template-columns: 200px 1fr;
    grid-template-rows: 80px 1fr 80px;
    grid-template-areas: "header header"
                         "left-section content"
                         "footer footer";
  }
}
```

Zdroj: vlastní

V dalším kroku budou umístěny položky do mřížky. K tomu bude využita vlastnost *grid-area* s nastavením příslušné hodnoty podle předchozího definováni vlastnosti *grid-template-areas*. Pro ukázku bude u každé sekce nastavené pozadí.

Zdrojový kód 21 Přiřazení sekcí do mřížky

```
.header {
  background: #EC8F5E;
  grid-area: header;
}

.left-section {
  background: #F1EB90;
  grid-area: left-section;
}

.content {
  background: #F3B664;
  grid-area: content;
}

.footer {
  background: #9FBB73;
  grid-area: footer;
}
```

Zdroj: vlastní

4.4.3 Srovnání implementací

Tabulka 3 Srovnání implementací rozložení s levým bočním panelem

Hodnocené kritérium	Flexbox	Grid
Struktura HTML	Elementy jsou na stejné úrovni	Elementy jsou na stejné úrovni
Čitelnost kódu	Definování CSS vlastností pro jednotlivé sekce, vyžaduje úpravu na více místech	Jednotná definice rozložení pomocí vlastnosti *grid-template-areas* nabízí čitelný CSS kód
Nastavení velikosti sekcí	Použití funkce calc(), velikosti jsou definovány u CSS tříd sekcí	Velikost sekcí je řízena nastavením velikosti mřížky (vlastnosti *grid-template-columns* a *grid-template-rows*)

Zdroj: vlastní

Zdrojový kód HTML byl v obou případech totožný a jednotlivé oblasti se nacházejí na stejné úrovni.

V případě modulu grid bylo využito vlastnosti *grid-template-areas*, která umožňuje definování mřížky pomocí názvu oblastí, které jsou označeny vlastnosti *grid-area*. Implementace pomocí gridu nabízí jednodušší správu velikostí jednotlivých oblastí pomocí vlastností grid-template-columns a grid-template-rows.

V případě řešení s pomocí modulu flexbox jsou velikosti definovány u elementů. Například pro šířku bočního panelu je velikost dopočítávána pomocí CSS funkce *calc()*. Definování velikostí na více místech a nutnost výpočtu snižuje čitelnost výsledného zdrojového kódu. V případě budoucí úpravy je pro vývojáře náročnější nalézt místo, kde změnu provést, a tak se zvyšuje i možnost případné chyby.

Pro rozložení s levým bočním panelem je vhodnější volit modul grid, jelikož nabízí možnost definování jednotlivých oblastí a možnost správy velikostí na jednom místě.

4.5 Tvorba karty produktu

Karta produktu představuje vizuální a informační prvek v seznamu produktů za účelem prezentace konkrétní položky. Karta je obvykle umístěna v řadě s dalšími

produkty. Poskytuje uživatelům rychlý přehled o důležitých informacích týkajících se daného výrobku. Na kartě mohou být zahrnuty klíčové údaje jako například název produktu, obrázek, cena, krátký popis a případně hodnocení produktu od zákazníků v podobě hvězdiček.

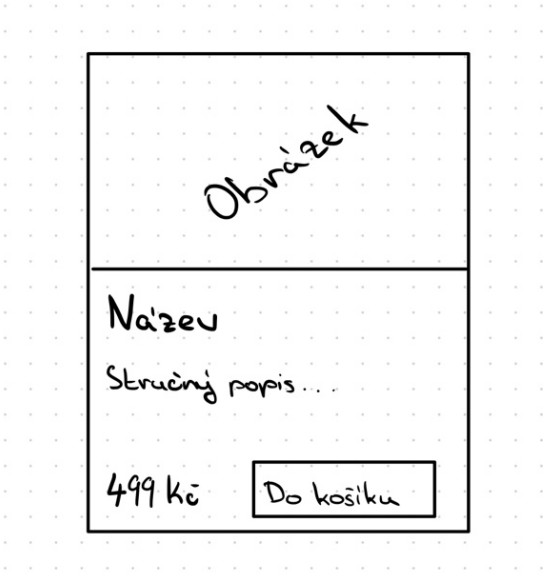
Uživatelé mohou interagovat s kartou interagovat v podobě přechodu na detailní stránku produktu nebo přidání položky do nákupního košíku.

Karta produktu má klíčový význam při usnadňování rozhodování zákazníků a vylepšuje uživatelskou interakci při prohlížení nabídky v internetovém obchodě.

4.5.1 **Návrh**

V příkladu bude implementována karta, jež bude obsahovat klíčové informace o produktu. Tato karta bude zahrnovat vizuální prvek v podobě obrázku produktu, název, stručný popis a tlačítko, které umožní uživateli přidat danou položku do nákupního košíku. Návrh bude obsahovat jednotný vizuál pro mobilní i desktopovou verzi, jelikož karta produktu má odlišné umístění v závislosti na velikosti obrazovky. Na desktopu je součástí řady s ostatními produkty, zatímco na mobilu se nachází samostatně.

Obrázek 30 Návrh karty produktu



Zdroj: vlastní

4.5.2 **Implementace pomocí Tailwind CSS**

Na začátku bude definován zdrojový kód HTML. Ten se bude skládat z hlavního HTML elementu *article,* uvnitř kterého bude značka *img* pro obrázek a blok *div*, který

bude obsahovat značku *h5* pro název produktu. Součástí bloku bude také odstavec textu a další blok, který bude seskupovat cenu a tlačítko pro přidání položky do košíku.

Tailwind CSS využívá třídy obsahující definici stylů, které se kombinují k vytvoření požadovaného vizuálního vzhledu. Kombinací těchto tříd lze dosáhnout libovolné požadovaného vzhledu.

Pro získání orámování kolem karty bude aplikována třída s názvem *border* a třída *border-black*, která aplikuje černou barvu orámování. Dále bude omezena šířka karty na 20 rem pomocí třídy *max-w-80*.

Na blok seskupující název, stručný popis a cenu s tlačítkem bude aplikována třída *flex*, která nastaví vlastnost *display* na hodnotu *flex*. Pro řazení položek uvnitř bloku bude přidána třída *flex-col*, která nastavuje vlastnost *flex-direction* na hodnotu *column*. Pro nastavení mezery mezi položkami bude přidána třída *gap-3*, která přidá vlastnost *gap* s hodnotou *0.75rem*.

4.5.3 **Implementace pomocí knihovny Bootstrap**

Implementace pomocí knihovny Bootstrap bude obsahovat stejný zdrojový kód jako v přechozím příkladu.

Pro splnění návrhu bude využito komponenty s názvem *card*, která nabízí předdefinovaný vzhled obsahující orámování s možností rozdělení obsahu. Pro aplikování stylů bude na HTML element *article* nastavena třída card. Následně bude *div*, který obaluje obsah aplikována třída *card-body*. Uvnitř bude element *h5* obsahovat CSS třídu *card-title* a popis produktu v elementu *p* bude obsahovat třídu *card-text*. Pro nastavení zobrazení rozložení ceny a tlačítka přidání do košíku bude zvolena kombinace tříd, které obsahuje Bootstrap k práci s modulem flexbox.

Pro dosažení obdobného finálního výsledku jako v případě implementace pomocí Tailwind CSS budou upraveny některé CSS třídy. Například pro nastavení barvy tlačítka bude na třídě *btn* nastavena barva orámování a barva pozadí. Na třídě *card-body* bude vlastnost *padding* upravena na hodnotu *0,5 rem*.

4.5.4 **Srovnání implementací**

Tabulka 4 Srovnání implementací karty produktu

Hodnocené kritérium	**Tailwind CSS**	**Bootstrap**
Rychlost vývoje	Vyšší časová náročnost, je třeba implementovat komponenty od základu	Jednodušší implementace, nabízí možnost využití předpřipraveného kódu jednotlivých komponent
Možnosti úprav vzhledu	Flexibilní, možnost kombinace utility tříd	Definováním vlastních CSS nebo přepsání výchozích CSS tříd knihovny Bootstrap

Zdroj: vlastní

Tailwind CSS umožňuje vytváření vlastního návrhu pomocí kombinace utility tříd, které přinášejí vysokou míru flexibility. Implementace základní kostry komponenty v případě knihovny Tailwind CSS trvá déle než u knihovny Bootstrap. Při následné úpravě k dosažení požadovaného vzhledu nabízí Tailwind CSS jednodušší systém, zatímco Bootstrap vyžaduje nutnost úpravy definicí tříd vlastními přidanými styly.

Pro tvorbu webových stránek, které vyžadují rychlou tvorbu s využitím předpřipravených komponent, je vhodná knihovna Bootstrap. Tailwind CSS je vhodný pro projekty, které vyžadují velkou flexibilitu v detailu a umožňují detailní úpravy výsledného vzhledu.

4.6 **Implementace webové stránky**

V této části práce bude implementována webová stránka přehledu produktů. Stránka bude obsahovat responzivní komponenty z předchozích příkladů.

Nejprve bude vytvořen zdrojový kód HTML. K tomu bude využito rozložení s levým bočním panelem. V horní části bude umístěn zdrojový z navigačního menu. V levém bočním panelu budou implementovány filtry.

Uvnitř HTML značky *section* bude umístěn nadpis pomocí elementu *h2*. Následně budou vytvořeny elementy *div*, které budou obalovat jednotlivé sekce filtrů. Každý filtr se bude skládat z názvu, který bude uvnitř značky *span* a jednotlivých možností, které budou zabaleny do elementu *div*. Každá možnost ve filtru bude pomocí značky *label* a *input*, který bude mít typ *checkbox*.

Zdrojový kód 22 Zdrojový kód HTML – Filtry

```
<section class="left-section filters">
  <h2>Filtry</h2>
  <div class="filter-section">
    <span class="filter-title">Kategorie</span>
    <div class="filter-options">
      <label><input type="checkbox">Keramické</label>
      <label><input type="checkbox">Plastové</label>
      <label><input type="checkbox">Ocelové</label>
    </div>
  </div>
  ...
</section>
```

Zdroj: vlastní

Pro nastavení vzhledu stránky bude využito kombinace knihovny Bootstrap a vlastních kaskádních stylů. Produktová karta bude implementována pomocí připravené komponenty *card* v knihovně Bootstrap.

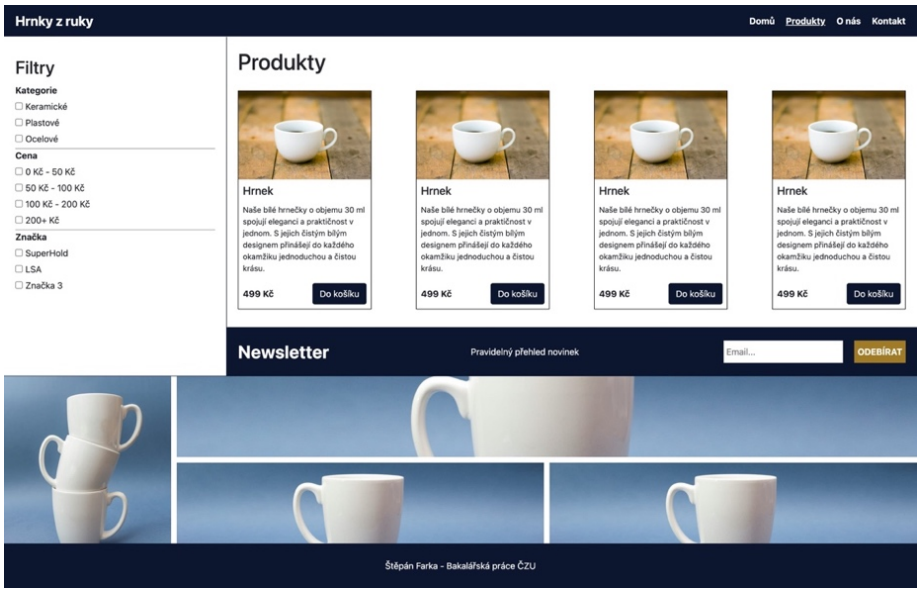
Primární barva stránky bude *#0c172f*, která se bude nacházet v navigačním menu, tlačítku pro přidání položky do košíku, patičce a části pro sběr emailů. Zdrojové kódy kaskádních stylů budou převzaty z předchozích příkladů a modifikovány za účelem sjednocení vzhledu.

V horní části obrazovky bude umístěno navigační menu. Aktivní odkaz bude zvýrazněn pomocí vlastnosti *border-bottom*.

Filtry budou od sebe odsazeny pomocí nastavení vlastnosti *border-bottom* na hodnotu *1px solid gray* a nastavení vnitřního odsazení vlastností *padding*.

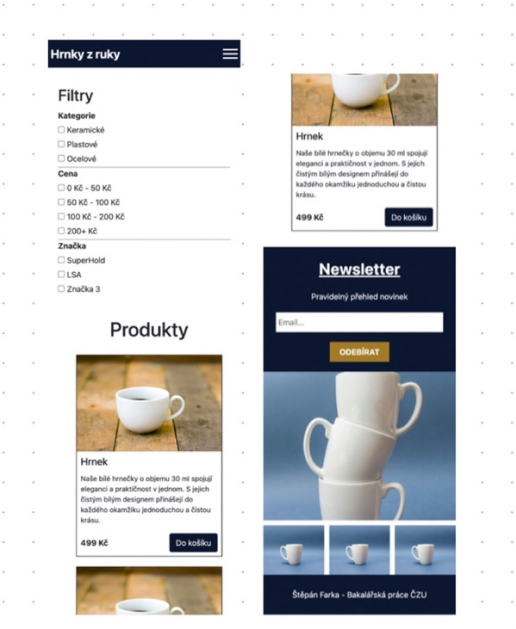
Ve spodní části obrazovky se bude nacházet produktová galerie, která bude implementována pomocí modulu grid.

Obrázek 31 Výsledný vzhled webové stránky na desktopu



Zdroj: vlastní

Obrázek 32 Výsledný vzhled webové stránky na mobilu



Zdroj: vlastní

4.6.1 Zhodnocení implementace

V příkladu byla vytvořena webová stránka, která splňuje požadavky responzivní webové stránky. Na mobilním zařízení je filtrovací část umístěna v horním části obrazovky pomocí rozdělení stránky s levým bočním panelem. Produkty jsou na obrazovkách s větší

šířkou rozloženy vedle sebe, zatímco na mobilním zařízení jsou umístěny pod sebe. Toho je dosaženo pomocí modulu flexbox a vlastnosti *flex-wrap*.

Část pro sběr emailů od zákazníků je implementována pomocí Container Queries, které se jsou vhodné pro tento typ komponent. Implementace v rámci webové stránky má následující chování:

- šířka (X až 829 px) - kompaktní zobrazení
- šířka (830 px až 899 px) - širší zobrazení
- šířka (900 px až 1229 px) - kompaktní zobrazení
- šířka (1230 px až X) - širší zobrazení

Tohoto chování lze dosáhnout právě pomocí Container Queries, jelikož způsob, jakým se komponenta zobrazí je definován množstvím dostupného prostoru.

5 Zhodnocení výsledků

Výsledkem bakalářské práce je analýza nástrojů a způsobů pro tvorbu responzivních webových stránek s následnou implementací webové stránky, která splňuje podmínky responzivního návrhu. Stránka se skládá z jednotlivých komponent, jež byly navrženy, implementovány a zhodnoceny.

Komponenty využívají moderní přístup k vytváření responzivních designů. Kombinují moduly flexbox a grid, které jsou dnes standardem pro tvorbu responzivních webových stránek a obsahují techniky Container Queries a Media Queries. Tento přístup umožňuje snadné a flexbilní přizpůsobení rozhraní podle různých typů zařízení a velikostí obrazovky. Výsledek jsou komponenty, které poskytují uživatelům optimální uživatelský zážitek, bez ohledu na to, jakým způsobem k obsahu přistupují.

Na základě srovnání s hodnotícími kritérii modulu flexbox a grid, je modul grid vhodnější pro tvorbu responzivních komponent s komplexnějším rozložením. Umožňuje tvorbu mřížky pomocí vlastnosti *grid-template-areas*. Jedná se o zápis, který nabízí možnost strukturovat zápis rozložení stránky do jednotlivých oblastí. V takových případech je výsledný zdrojový kód lépe čitelný a snadno spravovatelný. Díky pojmenovaným oblastem se lépe pracuje s rozložením stránky a manipulováním se strukturou mřížky. Flexbox je vhodný pro rozložení prvků v rámci jedné dimenze (řádek nebo sloupec), zatímco modul grid nabízí možnost správy prvků ve dvoudimenzionálním prostoru (řádky a sloupce). Modul flexbox je vhodný pro zarovnávání prvků. Nabízí snadné nastavení pro horizontální či vertikální centrování obsahu. V kontextu moderního webového návrhu je důležité brát v úvahu podporu prohlížečů, jelikož flexbox a grid se mohou chovat různě v jednotlivých prostředích a starší verze prohlížečů nemusí podporovat pokročilé funkcionality.

Dle výsledů SWOT analýzy v příkladu implementující techniku Container Queries, lze konstatovat, že Container Queries přináší výhodu vysoké flexibility responzivního návrhu, jelikož umožňují definovat styly na základě dostupných rozměrů komponenty, nikoli pouze rozměrech aktuálního okna prohlížeče. Tato flexibilita přináší designerům a vývojářům lepší možnosti pro přizpůsobení vzhledu a chování komponent různým zařízením, což v důsledku vede ke zlepšení uživatelské zkušenosti. Dále je technika vhodná pro typy komponenty, které mohou být vloženy nezávisle v různých částech webu. Příkladem může být komponenta pro sběr emailových adres klientů za účelem následného

marketingového sdělení, která se obvykle zobrazuje na více místech webové stránky. Container Queries mohou usnadnit vývoj PWA (Progressive Web Application) aplikací. Dotazy nabízejí aplikaci flexibilně reagovat na různé okolní faktory, jako například velikost obrazovky nebo orientace zařízení. Tímto způsobem se PWA může efektivně přizpůsobovat různým podmínkám uživatele a poskytovat optimální uživatelskou zkušenost.

Při porovnání hodnotících kritérií v příkladu implementace karty produktu, autor došel k výsledku, že Bootstrap je knihovna, která obsahuje sadu předdefinovaných komponent, jako například karta, navigační menu nebo formuláře, zatímco Tailwind CSS nabízí odlišný přístup k tvorbě uživatelského rozhraní a vyžaduje tvorbu celých komponent s využitím systému předefinovaných tříd. Jedním z klíčových rozdílů je skutečnost, že Tailwind CSS nevyžaduje použití předpřipravených komponent, a tak umožňuje vývojářům mít větší kontrolu nad detailem rozhraní. Mohou snadněji upravovat výsledný zdrojový kód dle svých potřeb, aniž by byly nuceni spoléhat na předem definované struktury, jako v případě Bootstrapu. Tailwind CSS nabízí větší flexibilitu a kontrolu, což znamená i vyšší míru abstrakce a složitějšího kódování, zatímco v případě Bootstrapu je vývoj, díky svým předdefinovaným komponentám, rychlejší. Pro úpravu vzhledu v knihovně Bootstrap je potřeba přidat další CSS třídy nebo předefinovat výchozí nastavení.

V důsledku lze konstatovat, že volba mezi knihovnou Bootstrap a Tailwind CSS je otázkou preferencí, finančních možností, konkrétních potřeb a úrovně detailu při tvorbě uživatelské rozhraní.

6 Závěr

V teoretické části byly pomocí literární rešerše odborných informačních zdrojů představeny nástroje a způsoby pro tvorbu responzivních webových stránek. Znalosti z teoretické části byly aplikovány při zpracování vlastní práce.

Praktická část byla zaměřena na porovnání způsobů a technik tvorby responzivních webových stránek použitím metody srovnání se stanovenými kritérii a SWOT analýzy s následnou implementací webové stránky. Ze zpracování praktické části byly formulovány výsledky.

Na základě srovnání se stanovenými kritérii modulu flexbox a grid je vhodné pro komplexní rozložení volit modul grid. Nabízí lepší čitelnost a správu výsledného kódu. V případě jednodimenzionálního prostoru postačuje použití modulu flexbox. Je třeba zdůraznit, že při výběru modulu flexbox nebo grid je vhodné zvážit konkrétní požadavky a strukturu webového návrhu, aby byla zvolena nejefektivnější možnost pro daný projekt.

Vzhledem k SWOT analýze je patrné, že technika Container Queries nabízí významné výhody při tvorbě responzivních komponent. Její využití je zejména vhodné při v případě komponent, které jsou navrhovány s ohledem na flexibilitu umístění v různých částech webové stránky. Container Queries umožňují přizpůsobit vzhled na základě dostupného místa, což přispívá lepší uživatelské zkušenosti.

Po provedení srovnání se stanovenými kritérii knihoven Tailwind CSS a Bootstrap je patrné, že Bootstrap je optimální volbou v případě projektů, které kladou důraz na rychlý vývoj při nízkých nákladech. Tailwind CSS je vhodný u komplexních projektů s vysokými nároky na vzhled.

Z bakalářské práce lze odvodit několik klíčových poznatků a závěrů o responzivních webových stránkách. Jedním z nich je důležitost responzivního návrhu v dnešním prostředí, kdy uživatelé přistupují na web z různých zařízení. Tento faktor vyžaduje po vývojářích důraz na implementaci webových stránek, které automaticky přizpůsobují vzhled rozlišení dané obrazovce. Dalším poznatkem je vhodnost použití kombinace technik Media a Container Queries, které umožňují efektivní dosažení responzivního vzhledu pomocí dotazů na šířku obrazovky nebo velikosti dostupného místa kontejneru.

Celkově lze z bakalářské práce vyvodit, že responzivní design je nedílnou součástí moderních webových stránek a jeho úspěšná implementace vyžaduje kombinaci správných technik a nástrojů pro dosažení požadovaného výsledku.

7 Seznam použitých zdrojů

[1] BOHYUN, Kim. Responsive Web Design, Discoverability, and Mobile Challenge. *ALA TechSource* [online]. 2013 [cit. 2023-08-27]. Dostupné z: https://www.journals.ala.org/index.php/ltr/article/view/4507

[2] HORIACHKO, Andrii. Web Development: Mobile First or Desktop First? *Softermii* [online]. [cit. 2023-08-27]. Dostupné z: https://www.softermii.com/blog/web-development-mobile-first-or-desktop-first

[3] CECI, Laura. Mobile internet usage worldwide - Statistics & Facts. *Statista* [online]. 2024 [cit. 2024-01-28]. Dostupné z: https://www.statista.com/topics/779/mobile-internet/#topicOverview

[4] Desktop vs Mobile vs Tablet Market Share Worldwide. *Statcounter* [online]. c2024 [cit. 2024-03-03]. Dostupné z: https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/worldwide/#monthly-200901-202312

[5] Desktop vs Mobile vs Tablet Market Share Asia. *Statcounter* [online]. c2024 [cit. 2024-03-03]. Dostupné z: https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/asia/#monthly-200901-202312

[6] MICHÁLEK, Martin. Co je to „Mobile First“? Ale doopravdy. *Vzhůru dolů* [online]. 2015 [cit. 2023-08-27]. Dostupné z: https://www.vzhurudolu.cz/prirucka/mobile-first

[7] BOSE, Shreya. Breakpoints for Responsive Web Design in 2023. *BrowserStack* [online]. 2023 [cit. 2023-08-27]. Dostupné z: https://www.browserstack.com/guide/responsive-design-breakpoints

[8] RADZIK, Krzysztof. What is mobile-first design? *Boldare* [online]. 2022 [cit. 2023-08-27]. Dostupné z: https://www.boldare.com/blog/what-is-mobile-first-design-strategy/

[9] O'LEARY, Rob. CSS breakpoints for responsive design. *LogRocker* [online]. 2023 [cit. 2023-08-27]. Dostupné z: https://blog.logrocket.com/css-breakpoints-responsive-design/

[10] Basic concepts of flexbox. *MDN Web Docs Mozilla* [online]. [cit. 2023-08-27]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Basic_Concepts_of_Flexbox

[11] COYIER, Chris. A Complete Guide to Flexbox. *CSS-Tricks* [online]. 2022 [cit. 2023-08-27]. Dostupné z: https://css-tricks.com/snippets/css/a-guide-to-flexbox/

[12] Backwards compatibility of flexbox. *MDN Web Docs Mozilla* [online]. [cit. 2023-08-27]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_flexible_box_layout/Backwards_compatibility_of_flexbox

[13] MICHÁLEK, Martin. *CSS: Moderní layout*. Martin Michálek - VzhůruDolů.cz, 2022. ISBN 978-80-88253-07-5.

[14] MICHÁLEK, Martin. CSS vlastnost flex-flow: zkratka pro určení směru a zalamování flexboxu. *Vzhůru dolů* [online]. 2021 [cit. 2023-08-27]. Dostupné z: https://www.vzhurudolu.cz/prirucka/css-flex-flow

[15] Flex-basis. *MDN Web Docs Mozilla* [online]. 2024 [cit. 2023-08-27]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/CSS/flex-basis

[16] MICHÁLEK, Martin. CSS vlastnost flex-basis: velikost položky flexboxu. *Vzhůru dolu* [online]. [cit. 2024-03-03]. Dostupné z: https://www.vzhurudolu.cz/prirucka/css-flex-basis

[17] Flex-grow. *MDN Web Docs Mozilla* [online]. 2024 [cit. 2024-03-09]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/CSS/flex-grow

[18] MICHÁLEK, Martin. CSS vlastnost flex-shrink: faktor smršťování položky flexboxu. *Vzhůru dolu* [online]. [cit. 2024-03-03]. Dostupné z: https://www.vzhurudolu.cz/prirucka/css-flex-shrink

[19] MICHÁLEK, Martin. CSS Grid: Kompletní příručka všech vlastností (s příklady). *Vzhůru dolů* [online]. 2019 [cit. 2023-08-27]. Dostupné z: https://www.vzhurudolu.cz/prirucka/css-grid

[20] *Can I use* [online]. [cit. 2024-03-03]. Dostupné z: https://caniuse.com

[21] HOUSE, Chris. A Complete Guide to CSS Grid. *CSS-Tricks* [online]. 2023 [cit. 2023-08-27]. Dostupné z: https://css-tricks.com/snippets/css/complete-guide-grid/#aa-special-units-functions

[22] MICHÁLEK, Martin. Vlastnosti grid-template-rows a grid-template-columns: definice explicitního gridu. *Vzhůru dolů* [online]. 2019 [cit. 2023-08-27]. Dostupné z: https://www.vzhurudolu.cz/prirucka/css-grid-template-rows-columns

[23] SEYEDI, Mojtaba. Grid-template-areas. *CSS Tricks* [online]. 2022 [cit. 2023-08-27]. Dostupné z: https://css-tricks.com/almanac/properties/g/grid-template-areas/

[24] MICHÁLEK, Martin. CSS3 Media Queries. *Vzhůru dolů* [online]. 2013 [cit. 2023-08-27]. Dostupné z: https://www.vzhurudolu.cz/prirucka/css3-media-queries

[25] MDN WEB DOCS, Mozilla. *Beginner's guide to media queries* [online]. [cit. 2024-01-28]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Media_queries

[26] MICHÁLEK, Martin. *Vzhůru do (responzivního) webdesignu*. Martin Michálek - VzhůruDolů.cz, 2018. ISBN 978-80-88253-00-6.

[27] SCHWARZ, Daniel. CSS Media for Printable Webpages. *Sympli* [online]. 2020 [cit. 2024-01-28]. Dostupné z: https://sympli.io/blog/a-quick-guide-to-css-for-printable-webpages

[28] Prefers-color-scheme. MOZILLA. *MDN Web Docs* [online]. c1998–2024 [cit. 2024-01-28]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/CSS/@media/prefers-color-scheme

[29] MICHÁLEK, Martin. Container Queries přicházejí. *Vzhůru dolů* [online]. 2023 [cit. 2023-07-30]. Dostupné z: https://www.vzhurudolu.cz/prirucka/container-queries

[30] CSS container queries. *MDN Web Docs Mozilla* [online]. [cit. 2023-08-27]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_container_queries

[31] BONAVENTURE, Onuorah. A guide to CSS container queries. *LOG ROCKET* [online]. 2023 [cit. 2024-01-28]. Dostupné z: https://blog.logrocket.com/css-container-queries-guide/

[32] KRAVETS, Una. Container queries land in stable browsers. *Web.dev* [online]. 2023 [cit. 2023-08-27]. Dostupné z: https://web.dev/cq-stable/

[33] ELEMUWA, Fimber. When and how to choose between media queries and container queries. *LogRocker* [online]. 2022 [cit. 2023-08-27]. Dostupné z: https://blog.logrocket.com/choose-between-media-container-queries/

[34] What Is Bootstrap? An Introduction to the Popular Web Development Framework. *Altcademy* [online]. 2023, s. 1 [cit. 2023-08-05]. Dostupné z: https://www.altcademy.com/blog/what-is-bootstrap-an-introduction-to-the-popular-web-development-framework/

[35] Grid system. *Bootstrap* [online]. [cit. 2023-08-27]. Dostupné z: https://getbootstrap.com/docs/4.0/layout/grid/

[36] Alerts. *Bootstrap* [online]. [cit. 2023-08-27]. Dostupné z: https://getbootstrap.com/docs/4.0/components/alerts/

[37] DE ROY, Soham. What is Tailwind CSS? A Beginner's Guide. *FreeCodeCamp* [online]. 2022 [cit. 2023-08-27]. Dostupné z: https://www.freecodecamp.org/news/what-is-tailwind-css-a-beginners-guide/

[38] INALA, Prahlad. History of Tailwind CSS. *PRAHLAD INALA* [online]. 2023 [cit. 2023-08-27]. Dostupné z: https://blogs.prahladinala.in/history-of-tailwind-css

[39] MICHÁLEK, Martin. Utility CSS: K čemu jsou dobré systémy jednoúčelových tříd? *Vzhůru dolů* [online]. 2019 [cit. 2023-08-27]. Dostupné z: https://www.vzhurudolu.cz/prirucka/css-utility

[40] Utility-First Fundamentals. *Tailwind CSS* [online]. [cit. 2023-08-27]. Dostupné z: https://tailwindcss.com/docs/utility-first

[41] Configuration. *Tailwind CSS* [online]. [cit. 2023-08-27]. Dostupné z: https://tailwindcss.com/docs/configuration

[42] IHECHIKARA, Abba. How to Use Tailwind CSS to Rapidly Develop Snazzy Websites. *Kinsta* [online]. 2023 [cit. 2023-08-27]. Dostupné z: https://kinsta.com/blog/tailwind-css/

[43] Populating the page: how browsers work. *MDN Web Docs Mozilla* [online]. [cit. 2024-01-25]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/Performance/How_browsers_work

[44] KONEČNÝ, Ondřej. Jak funguje CSS Box Model. *Frontend Garden* [online]. 2020 [cit. 2024-01-29]. Dostupné z: https://frontend.garden/clanky/jak-funguje-css-box-model/

[45] The box model. *MDN Web Docs Mozilla* [online]. c2024 [cit. 2024-03-03]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/The_box_model

[46] KONGER, Snehasish. How browsers work | The Rendering Engine. *Scientyfic World* [online]. 2023 [cit. 2024-01-25]. Dostupné z: https://scientyficworld.org/how-browsers-work-the-rendering-engine/

[47] Critical rendering path. *MDN Web Docs Mozilla* [online]. [cit. 2024-01-27]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/Performance/Critical_rendering_path

[48] GARSIEL, Tali. How browsers work. *Web dev* [online]. [cit. 2024-01-25]. Dostupné z: https://web.dev/articles/howbrowserswork#the_main_flow

[49] Populating the page: how browsers work. *MDN Web Docs Mozilla* [online]. [cit. 2024-01-27]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/Performance/How_browsers_work

[50] REGONDA, Akhil. The Anatomy of Browser Rendering: How Web Pages Come to Life. *Medium* [online]. 2023 [cit. 2024-01-28]. Dostupné z: https://medium.com/@regondaakhil1509/the-anatomy-of-browser-rendering-how-web-pages-come-to-life-6fa9e801a3f

8 Seznam obrázků, tabulek, grafů a zkratek

8.1 Seznam obrázků

8.2 Seznam tabulek

8.3 Seznam zdrojových kódů

8.4 Seznam použitých zkratek

CSS - Cascading Style Sheets

CSSOM - CSS Object Model

DOM - Document Object Model

HTML – HyperText Markup Language

PWA – Progressive web app

SEO – Search engine optimization

Přílohy

Vytvořené zdrojové kódy jsou dostupné v přiloženém .zip souboru či na poskytnutém paměťovém médiu.