Pedagogická fakulta
Faculty of Education

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice


**Jihočeská univerzita v Českých Budějovicích**

Pedagogická fakulta

Katedra informatiky


# Uživatelská rozhraní webu s využitím Utility CSS a frameworku Tailwind
# Web user interfaces using the CSS Utility and the Tailwind framework

Bakalářská práce


**Vypracoval:** David Král

**Vedoucí práce:** PaedDr. Petr Pexa, Ph.D.



České Budějovice 2020

# JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH

Pedagogická fakulta

Akademický rok: 2019/2020

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

| | |
|---|---|
| Jméno a příjmení: | **David KRÁL** |
| Osobní číslo: | **P18479** |
| Studijní program: | **B7507 Specializace v pedagogice** |
| Studijní obor: | **Informační technologie a e-learning** |
| Téma práce: | **Uživatelská rozhraní webu s využitím Utility CSS a frameworku Tailwind** |
| Zadávající katedra: | **Katedra informatiky** |

## Zásady pro vypracování

Cílem bakalářské práce je zpracovat problematiku Utility CSS s využitím Utility-first CSS frameworku Tailwind. V Utility-first CSS jde o psaní kódu pomocí tříd, které mají jen jeden účel, jedná se tedy o celkový přístup k tvorbě kódu, postaveném na jednotlivých třídách, tzv. „utility classes". Téměř vždy jde o kombinaci CSS vlastnosti a nějaké její hodnoty, výhodou je neměnnost a nezávislost na kontextu. Nástroj Tailwind usnadňuje vývoj UI designu, kdy ale není nutně potřebná znalost Utility CSS, bude tedy porovnána tvorba CSS kódu s využitím frameworku a klasickým postupem. Obsahem teoretické části bude představení Utility CSS, frameworku Tailwind a jeho porovnání s frameworkem Bootstrap, praktická část bakalářské práce bude založena na tvorbě vlastní webové aplikace, kde budou Utility CSS a framework detailně představeny. Výsledkem práce bude porovnání efektivity, časové náročnosti a čitelnosti kódu, zajímavostí bude ovládání uživatelských formulářů (např. vyhledávání), na kterých bude aplikována aktuální blind friendly technika speech recognition (rozpoznávání hlasu a převodu na text).

| | |
|---|---|
| Rozsah pracovní zprávy: | **40** |
| Rozsah grafických prací: | **CD ROM** |
| Forma zpracování bakalářské práce: | **tištěná** |

Seznam doporučené literatury:


1. MICHÁLEK, Martin. Vzhůru dolů – webová kodéřina ze všech stran [online]. [cit. 2020-04-04]. Dostupné z: https://www.vzhurudolu.cz/prirucka/css-utility
2. WATHAN, Adam a Steve SCHOGER. Tailwind CSS – A Utility-First CSS Framework for Rapidly Building Custom Designs [online]. [cit. 2020-04-04]. Dostupné z: https://tailwindcss.com
3. Joyent, Inc. Node.js [online]. [cit. 2020-04-04]. Dostupné z: https://nodejs.org/en/
4. OTTO, Mark a Jacob THORNTON. Bootstrap ? The most popular HTML, CSS, and JS library in the world. [online]. [cit. 2020-04-04]. Dostupné z: https://getbootstrap.com/
5. SpeechRecognition - Web APIs | MDN [online]. [cit. 2020-04-04]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/SpeechRecognition


Vedoucí bakalářské práce: **PaedDr. Petr Pexa, Ph.D.**
Katedra informatiky

Datum zadání bakalářské práce: **5. dubna 2020**
Termín odevzdání bakalářské práce: **30. dubna 2021**

L.S.

**doc. RNDr. Helena Koldová, Ph.D.**
děkanka

**doc. PaedDr. Jiří Vaníček, Ph.D.**
vedoucí katedry

V Českých Budějovicích dne 5. dubna 2020

# Prohlášení

Prohlašuji, že svoji bakalářskou práci na téma uživatelská rozhraní webu s využitím Utility CSS a frameworku Tailwind jsem vypracoval(a) samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

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

v Českých Budějovicích dne .....

David Král

# Abstrakt

Cílem bakalářské práce je zpracovat problematiku Utility CSS s využitím Utility-first CSS frameworku Tailwind. V Utility-first CSS jde o psaní kódu pomocí tříd, které mají jen jeden účel, jedná se tedy o celkový přístup k tvorbě kódu, postaveném na jednotlivých třídách, tzv. „utility classes“. Téměř vždy jde o kombinaci CSS vlastnosti a nějaké její hodnoty, výhodou je neměnnost a nezávislost na kontextu. Nástroj Tailwind usnadňuje vývoj UI designu, kdy ale není nutně potřebná znalost Utility CSS, bude tedy porovnána tvorba CSS kódu s využitím frameworku a klasickým postupem. Obsahem teoretické části bude představení Utility CSS, frameworku Tailwind a jeho porovnání s frameworkem Bootstrap, praktická část bakalářské práce bude založena na tvorbě vlastní webové aplikace, kde budou Utility CSS a framework detailně představeny. Výsledkem práce bude porovnání efektivity, časové náročnosti a čitelnosti kódu. Zajímavostí bude ovládání uživatelských formulářů (např. vyhledávání), na kterých bude aplikována aktuální blind friendly technika speech recognition (rozpoznávání hlasu a převodu na text).

# Klíčová slova

CSS, utility CSS, Tailwind, Bootstrap, front-end, framework, uživatelské rozhraní, webdesign, speech recognition

# Abstract

The point of the bachelor thesis is to process the issue of Utility CSS using the Utility-first CSS framework Tailwind. Utility-first CSS is about writing code using classes that have only one purpose, so it is a general approach to creating code, built on individual classes, the so-called "utility classes". It is almost always a combination of CSS properties and some of its value, the advantage is immutability and independence of context. The Tailwind tool facilitates the development of UI design, but when the knowledge of the CSS Utility is not necessary, so the creation of CSS code using the framework and the classic procedure will be compared. The content of the theoretical part will be the introduction of Utility CSS, Tailwind framework and its comparison with the Bootstrap framework, the practical part of the bachelor thesis will be based on creating your own web application, where Utility CSS and framework will be introduced in detail. The result of the work will be a comparison of efficiency, time and readability of the code. Of interest will be the control of user forms (eg search), on which the current blind friendly speech recognition technique (voice recognition and conversion to text) will be applied.



# Keywords

CSS, utility CSS, Tailwind, Bootstrap, front-end, framework, user interface, webdesign, speech recognition

# Poděkování

Tímto bych chtěl poděkovat vedoucímu mé bakalářské práce panu PaedDr. Petru Pexovi, Ph.D. za nápomocné rady, ochotu a vedení při zpracování mé závěrečné práce.

# Obsah

# 1 Úvod

## 1.1 Východiska práce

Při tvorbě a úpravě webových stránek se můžeme setkat s velikostně obrovskými, obsahově opakujícími se CSS soubory, které mohou brzdit rychlost, být nepřehledné nebo těžko spravovatelné. Problém, který se také může objevit, je nesmyslné pojmenování CSS tříd.

Utility CSS a framework Tailwind, který je na konceptu utility CSS postaven, přichází se snahou snížit velikost a zlepšit přehlednost stylů, aby nedocházelo k zbytečným duplicitám v kódu. Dalším přínosem by měla být znovupoužitelnost CSS tříd nezávisle na HTML elementu. Objevují se tedy unikátní identifikátory a zbavíme se nutnosti vymýšlet vypovídající názvy tříd nebo modifikátory pro jednotlivé HTML elementy.

Výhodu Tailwind frameworku vidím například v tom, že není primárně postaven na komponentách a pro jeho samotné využití nemusíme znát CSS jako takové. Při základním nastavení by měl tedy být použitelný i pro laiky, kteří se orientují pouze v HTML.

## 1.2 Cíle práce

Cílem bakalářské práce je představit utility CSS a framework Tailwind, který konceptu utility CSS využívá.

V teoretické části představím utility CSS, framework Tailwind, veškeré využité webové technologie a porovnám framework Tailwind s Bootstrapem.

V praktické části vytvořím několik jednotných komponent, na kterých Tailwind představím a porovnám jej s klasickým přístupem psaní CSS. Na závěr vytvořím responzivní webovou stránku s využitím utility CSS, Tailwind a popsaných technologií. Stránka bude sloužit jako prezentační pro představení jednoho z odvětví firmy Smart events.

## 1.3 Metody práce

V úvodu práce popíši využité webové technologie a detailněji představím utility CSS a framework Tailwind. Součástí bude porovnání frameworku Tailwind oproti Bootstrap.

Získané znalosti využiji pro tvorbu ukázkových komponent a webové stránky. Výsledná webová stránka bude responzivní, vznikne tedy verze pro desktopy, tablety i mobilní zařízení. Na závěr provedu ještě porovnání tvorby pomocí frameworku Tailwind oproti klasickému přístupu.

# 2 HTML

HTML (HyperText Markup Language) je nejzákladnějším prvkem pro tvorbu webových stránek. Pomocí tohoto jazyka určíme strukturu webu. Pro tvorbu webů se využívá v kombinaci s jinými technologiemi, které se starají o stylizaci (CSS) a funkční část (JavaScript). Z názvu technologie lze vytáhnout dva pojmy, Hypertext a Markup. [1][2]

První slovo neboli Hypertext zastupuje propojení jednotlivých stránek s jinými. Může se jednat o interní spojení, tedy v rámci jednoho webu, nebo externí, kdy web odkazuje na jiný v internetu. [1][2]

Slovo Markup neboli označení znázorňuje využití značek, které slouží k tomu, aby prohlížeč věděl, jakým způsobem jednotlivé texty a obrázky interpretovat. HTML označení obsahuje speciální elementy, které jsou tvořeny pomocí tagů. [1][2]

Tagy obsahují název, případně atributy, a jsou ohraničené pomocí ostrých závorek, ve kterých se nachází například další HTML element, text nebo obrázek. Pokud se element skládá ze začátečního i koncového tagu, nazývá se tag párový. Příkladem párového tagu může být odstavec, nadpis nebo odkaz. Druhou variantou je tag nepárový, který nemá ukončující tag a ukončuje se pomocí lomítka před uzavírací závorkou. Jedním z nejpoužívanějších je obrázek nebo zalomení řádku. [1][2]

```
1 <p>Odstavec</p> <!-- parovy tag -->
2 <img src="/obrazek.jpg" alt="obrazek" /> <!-- neparovy tag -->
```

Příklad 1: Ukázka HTML elementů

## 2.1 HTML5

Vývoj specifikace (původně nazývána Web Applications 1.0) začal v roce 2004, pod křídly WHATWG a byl mimo organizaci W3C. Na vývoji se podílel Apple,

Mozilla Foundation a Opera Software. Jednalo se o snahu posunout HTML a vyhnout se tak technologii XHTML2. Autorská práva byla upravena tak, aby je vlastnili všechny tři zmíněné společnosti a specifikace byla znovupoužitelná. [3]

WHATWG byl založen na pár základních principech. Jedním z nich byl ten, že současné stránky musejí být v HTML5 validní. Druhý princip spočíval v interoperabilititě neboli jasně definované chování. Toto bylo konkrétně mířeno na tvůrce prohlížečů. Muselo být jasné, jak nové specifikace implementovat. Následující snahou bylo zajistit přístupnost, nehledě na mluvený jazyk nebo zařízení. Dalším požadavkem bylo obsáhnout předchozí specifikace. Konkrétně z HTML 4.01, XHTML 1.1 a DOM Level 2 HTML. To znamenalo překročit standard specifikací, který byl stanoven. [3][4]

### 2.1.1 HTML5 pod W3C

Doposud vše bylo mimo organizaci W3C, což se roku 2006 začalo měnit. Organizace projevila zájem o to, podílet se na vývoji. To vedlo k tomu, že roku 2007 vytvořila skupinu, která s WHATWG na vývoji spolupracovala. Nakonec bylo W3C umožněno specifikaci zveřejnit pod autorskými právy W3C, avšak na stránkách WHATWG zůstala verze s méně omezující licencí. [4]

Následujících několik let pracovaly skupiny společně pod stejným editorem lan Hicksonem. Roku 2011 si obě skupiny uvědomily rozdílnost cílů. W3C chtěly přestat s funkcemi pro HTML 5.0, zatímco WHATWG chtěli pokračovat v rozvíjení a vylepšování specifikace. W3C uprostřed roku 2012 představuje nový tým, který má na starosti vytváření HTML 5.0 a přípravu pracovních konceptů pro nové verze HTML. [4]

Od té doby si W3C vybírá nejlepší úpravy od WHATWG, které řeší zjištěné chyby ve specifikaci W3C HTML. HTML editoři z W3C zároveň přidali opravy, které vznikly z diskuze a rozhodnutí webové platfomy WG, a také opravili chyby, které nebyly sdíleny od WHATWG. [4]

# 3 CSS

CSS (Cascading Style Sheets) je jazyk, který slouží k popsání prezentace webových stránek, včetně barev, rozložení a fontů písma. Umožňuje nám adaptovat prezentaci na různá zařízení. Například pro desktop, tablet, mobilní zařízení nebo tiskárnu. CSS není přímo spojené s HTML a je možné jej využít s jakýmkoliv jazykem, který je založený na XML. [6]

Hlavní výhodou je možnost snadno spravovat celkový styl webu umístěním kódu do samostatného dokumentu. Každá stránka webu následně odkazuje při načítání zpět na CSS soubor, tedy styl stránky odpovídá zbytku webu.[6]

## 3.1 CSS3

CSS3 je nejnovější vývoj jazyka kaskádových stylů a slouží jako rozšíření CSS2.1, které je jeho základem. Hlavním přínosem bylo rozdělení CSS na moduly. Už se tedy nevyvíjelo jako celek, ale v samostatných modulech typu CSS Color nebo Media Queries. Každý modul přidává funkcionalitu a/nebo nahrazuje část specifikace CSS2.1. Vývoj CSS se proto přestal označovat verzí, jelikož různé moduly mohly být v odlišné verzi. Dejme tomu mohly vzniknout selektory úrovně 4 předtím, než se dokončí ostatní moduly úrovně 3. Mohou se objevit i moduly úrovně 1. To jsou takové, které nemají ekvivalent ve verzi 2 a jsou tedy zcela nové. Ty, které ekvivalent mají, navazují a začínají na stupni 3. [7][8]

Tento fakt zmiňují oficiální stránky w3c, avšak až u verze 4, kde se můžeme dočíst, že moduly mohou dosáhnout vyšší úrovně než 3, ale samotný jazyk CSS už nemá úroveň. CSS3 je používáno pouze jako termín k odlišení od předchozích monolitických verzí. [8]

V částech, které jsou prozatím experimentální, nalezneme vendor prefixy (například -moz- nebo -webkit-), a je potřeba používat je s opatrností, protože jejich syntaxe nebo sémantický význam se může v budoucnu změnit. [9][10]

Pokud se podíváme na specifické novinky, které v této verzi přibyly, nalez-

neme zde vlastnosti jako zaoblené rohy, stíny, gradienty pozadí, transformace, přechody, animace a spoustu dalšího. Moduly, kterým budu věnovat více pozornosti, se zasloužily o výrazné zjednodušení tvorby layoutů. [9][10]

### 3.1.1 Flexible Box Layout modul

Flexible box, zkráceně nazývaný flexbox, je specifikace pro návrh uživatelských rozhraní a rozšiřuje display vlastnost z CSS2. Jedná se o první layoutovací prostředek, který pro to byl vymyšlen. Předtím se využívaly různé techniky jako floaty, absolutní pozicování nebo vlastnost display: table. Všechny tyto metody však pro rozvržení nebyly na rozdíl od flexboxu primárně určeny. [11][12]

Hlavní přednost nalezneme již v názvu. Flexible lze přeložit jako pružný nebo flexibilní a přesně na tomto principu flexbox funguje. Tvoří jej takzvaný flex kontejner a flex položka, která je přímým potomkem kontejneru. Tato položka se může v kontejneru flexibilně zvětšovat či zmenšovat na základě volného místa v kontejneru. [11][12]

Flexbox se doporučuje využít spíše pro komponenty, které jsou uvnitř stránky než pro celostránkový layout. Samozřejmě to není zakázané, ale více doporučovaný je grid layout. Jedním důvodem je postupné vykreslování, které v kombinaci s flexboxem na pomalejším připojení způsobovalo vychýlení zarovnání a vodorovné posunutí. Dalším je možnost gridu definovat mřížku v obou směrech, tedy v řádcích i sloupcích [12][13][15]

Pokud se podíváme na podporu v prohlížečích, pak modul Flexible Box Layout je u nových verzí ve všech nejčastěji používaných browserech zcela podporován. Jediný problém se může objevit u Internetu Explorer, který podporu poskytuje pouze částečnou. [14]

### 3.1.2 Grid Layout modul

Grid, stejně jako flexbox, vznikl za účelem tvorby layoutů a rozšiřuje vlastnost display. Rozdíl spočívá v tom, že grid již nevyužívá flexibilitu v rámci prázd-

ného prostoru, ale definuje systém rozložení založený na pravidelné mřížce. Mřížce můžeme definovat jak řádky, tak sloupce, a proto je vhodná pro celostránkové a komplexní layouty.[15][16]

Vlastnosti, které mřížce můžeme nastavit, jsou různé. Jedná se například o explicitní definici mřížky, kde přesně stanovíme šířku jednotlivého sloupce nebo řádku. Pokud neznáme přesný počet HTML elementů, nabízí se nám i implicitní definice mřížky. To znamená, že stanovíme stejnou hodnotu pro všechny sloupce či řádky. Stejně jako u flexboxu můžeme určit pořadí prvků nebo si například vybrat konkrétní řádek nebo sloupec, kam bude prvek umístěn. Užitečným nastavením se také jeví velikost mezer mezi jednotlivými buňkami.[15]

U tohoto modulu nalezneme 3 úrovně, přičemž aktuálně se pomalu implementuje druhá. Ta přinesla do gridu novinku zvanou subgrid, pro vnořené mřížky. Tento koncept by měl umožnit sladit mřížku kontejneru a vnořeného elementu. Díky tomu se mřížky podřízeného elementu podílejí na změně velikosti nadřazené mřížky a to umožní zarovnat obsah obou mřížek. Poslední úroveň, která je zatím rozpracovaná a sbírá zpětnou vazbu, obohatí grid o rozvržení masonry (zdiva). Princip této novinky spočívá v tom, že mezery, které zanechají menší prvky v řádku/sloupci, vyplní element na následujícím řádku/sloupci. To znamená, že v rozložení nevzniknou mezery. Aktuálně toho lze docílit pomocí JavaScriptu a umožňuje to vytvořit kreativní a zajímavé galerie obrázků. Umím si představit, že by se s pomocí tohoto daly vytvořit i zajímavé přehledy projektů nebo sponzorů, kde se mohou lišit rozložení a velikosti log. Tato připravovaná novinka mi tedy přijde velmi zajímavá. [5][17][18][19]

Jak již bylo zmíněno, aktuálně má tento modul 3 úrovně. První úroveň je v nových verzích nejpoužívanějších prohlížečů podporována bez problémů, až na Internet Explorer, kde jako u flexboxu má částečnou podporu. Úroveň 2 zatím nalezneme pouze ve Firefoxu a poslední úroveň jen ve Firefoxu, verze Nightly. [19]

### 3.1.3 Multi-column Layout modul

Tento modul nám umožňuje element rozložit do více sloupců s mezerou a oddělovačem mezi nimi. Nejčastěji to lze využít při novinové sazbě, kde snadno rozložíme text do více sloupců.[21][22]

Modul nám umožňuje nastavit šířku sloupce, počet sloupců, šířku mezery mezi sloupci a specifikaci oddělující čáry. Pokud se dostaneme na menší rozlišení a dle definice šířky nebude pro další sloupec prostor, prohlížeč automaticky přestane vícesloupcové rozložení využívat.[21][22]

Tento modul má zatím spíše částečnou podporu v nových verzích nejpoužívanějších prohlížečů. Plnou podporu zatím nalezneme jen u Safari nebo Internetu Explorer. [23]

### 3.1.4 Box Alignment modul

Modul, který rozeberu jako poslední, obsahuje funkce pro zarovnání boxů v jejich kontejnerech. Lze jej využít u blokového, flexibilního, mřížkového nebo tabulkového rozložení. [25]

Umožňuje nám určit směr zarovnání (primární osou je obvykle horizontální, lze ji změnit), nebo co konkrétně zarovnáme. Pro zarovnání na hlavní ose máme k dispozici justify-* (justify-items, justify-content a jiné), pro zarovnání na příčné ose align-* (align-items, align-content a jiné). [24]

Můžeme zarovnat všechny potomky, jednoho nebo obsah mezi nimi. Pro zarovnání položek lze využít vlastnost *-items (justify-items, align-items a jiné), pokud zarovnává sebe sama, pak jde o vlastnost *-self (justify-self, align-self a jiné), a poslední možností je distribuce obsahu *-content (justify-content, align-content a jiné). [24]

# 4 Utility CSS

Cílem tohoto přístupu je psát třídy, které mají pouze jeden záměr, takzvané „utility classes“. Může se jednat například o barvu pozadí, konkrétním příkladem může byt deklarace pro bílé pozadí:

```
1 .bg-white { background-color: white; }
```

Příklad 2: Ukázka utility CSS (pozadí)

Třída má vcelku vypovídající název o vlastnosti a hodnotě, kterou zastupuje. Pokud bychom tento přístup využili ve větším měřítku, nic nám nebrání nastylovat tímto způsobem celé komponenty. [26]

Pro tuto ukázku jsem si vybral příklad, který na svých stránkách uvádí framework Tailwind:

```
1 <div class="p-6 max-w-sm mx-auto bg-white rounded-xl shadow-md
     flex items-center space-x-4">
2   <div class="flex-shrink-0">
3     <img class="h-12 w-12" src="/img/logo.svg" alt="ChitChat Logo">
4   </div>
5   <div>
6     <div class="text-xl font-medium text-black">ChitChat</div>
7     <p class="text-gray-500">You have a new message!</p>
8   </div>
9 </div>
```

Příklad 3: Ukázka utility CSS (upozornění chatu) [27]

Příklad výše je horizontální komponenta, která znázorňuje upozornění chatu. Obsahuje na levé straně logo a na pravé straně větší text zastupující titulek a pod ním odstavec. Poprvé, když jsem se setkal s tímto způsobem zápisu

a zkoušel jsem si jej, přišel mi vcelku nepohodlný a zmatečný. Zkoušením a čtením článků jsem pomalu objevil výhody a vhodné použití této metody.

## 4.1 Adam Morse - Počátky utility-first CSS

Hlavní autor Tachyons, prvního utility-first CSS frameworku, Adam Morse již v roce 2016 popsal hlavní problémy rostoucí velikosti a těžké spravovatelnosti CSS. Narážel na problém nabalujících selektorů, aby bylo dosaženo větší váhy, které následně upravují pouze jednu vlastnost na jednu hodnotu. Popisoval to jako mentální model, kdy bylo snahou změnit jednu, nebo dvě věci v rozhraní, které byli v zadání a zároveň nerozbít nic jiného. [26][28]

Zmínil se také o své teorii, kde přemýšlel nad obousměrnou cestou informací. Tato teorie se zakládala na tom, že pokud si přečtu HTML, měl bych být schopen určit, co s ním CSS provede. Zároveň, pokud si přečtu CSS, budu schopen určit, co provede s HTML blokem.[28]

Jako ukázku uvedl následující:

```
1 .red {
2   color: #FF4136;
3 }
4
5 <div class="red">Some text</div>
```

Příklad 4: Adam Morse - identifikátor červeného písma [28]

V tomto příkladu, který nazval zmíněnou oboustrannou cestou informace, již můžeme vidět podstatu utility CSS. Nesetkal se s příliš pozitivním ohlasem ohledně názvu třídy, ale byl s tím spokojen. Z názvu a vlastnosti bylo jasné, že elementu změní barvu písma na červenou. Jediný případ, kdy mu připadalo toto označení špatné, bylo v případě, že přidělena barva je jiná než červená, nebo bude v budoucnu několikrát předefinována na jiné odstíny červené. [28]

Zároveň z HTML elementu bylo vidět, že je nějakým způsobem spojen

s červenou barvou. Dle mého názoru, z HTML nebylo zcela jasné, zda se jedná o změnu pozadí elementu nebo barvu textu, myšlenka však byla vypovídající. [28]

Současně se s tímto příkladem zmiňoval o snadnosti naučení skupin tříd, které dělají jednu věc správně a jsou znovupoužitelné. Tento koncept neměnnosti nepředstavoval novinku, ale nebyl v komunitě CSS využívaný. Demonstroval to na vzoru, kde vycházel z toho, že bychom měli funkci, která se nazývá velikostSouboru. Tato funkce přijímá jako parametr název souboru a vrací jeho velikost. Zní to velmi jednoduše a znovupoužitelně pro více souborů. Občasně by však tato funkce vrátila počet řádků souboru, což už není tak vyhovující. Tímto poukazoval na jakékoliv předefinování CSS třídy. [28]

S tímto problémem se setkával i u velkých úspěšných společností jako například GitHub nebo Adobe. Zmínil se v této souvislosti o principu DRY (don't repeat yourself, v překladu neopakuj sám sebe) a pozastavoval se nad tím, jestli opravdu tento model následují. Zda třídy a vlastnosti, které pomocí nich nastavují, jsou rozumně znovupoužitelné. Protože pokud bychom psali tak, že to někdo jiný po nás nedokáže použít, napíše si stejnou část kódu znovu a dostáváme se zpět k problému s duplicitami. [28]

Jak tedy definovat DRY v tomto kontextu? Nějakou duplicitu vždy budeme mít, ať už v CSS nebo HTML. Nehledě na dovednosti v CSS, není možné vytvořit novou komponentu bez editace HTML. Lze však vytvořit novou komponentu bez toho, abychom museli přidávat nové CSS. [28]

Pokud se opakujeme v HTML, neovlivní to velikost natolik, jako by tomu bylo u neustálého přidávání nových stylů. Uživatel si nemusí stahovat všechny HTML soubory, aby si prohlédl 1 stránku. V případě CSS však návrh bývá takový, že pro zobrazení 1 stránky jsou stažené CSS celého webu. [28]

Soubory, na které je tímto poukazováno jsou výsledkem toho, že tvůrci CSS generují dlouhé selektory, které zvyšují váhu kaskády a následně upravují pár věcí u konkrétního prvku. S postupem času je těchto malých změn potřeba udělat více a s tím roste váha selektorů, zvětšuje se velikost souboru a zároveň

se prodlužuje doba, kterou strávíme laděním CSS. Tento způsob zapříčiní, že refaktorování nebo odstraňování nepotřebného CSS se stane těžkým a časově náročným. Proto bude snazší vytvářet stále nové a nové styly. [28]

Tyto problémy mají vliv na škálovatelnost dvěma způsoby. Prvním z nich je náročnější práce na aplikaci, jelikož není šance zapamatovat si všechny selektory, které jsme nadefinovali. Pokud si je nepamatujeme, nebudeme je mít osvojené a s tím klesá pravděpodobnost, že bychom je znovu použili. To opět vede k přidávání nových stylů a narůstající velikosti obsahu, kterou musí uživatelé stáhnout a tvůrci spravovat. [28]

Přes všechna specifika problémů dosavadních CSS se dostal ke svému přirovnání k lego blokům. Zakládalo se na tom, že pokud při stavění z lega nalezl blok 4x1 v modré barvě, nespojil si ho s konkrétní částí stavby. Například, nejedná se o blok, který patří na zadní část letadla. Jedná se o blok, který lze použít kdekoliv, kde je potřeba a hodí se. Nezáleželo, zda byl blok použit pro stavbu auta nebo domu, ale na tom, že bylo jasné, jak daný blok využít. Blok se dal aplikovat na každou novou stavbu a kontext chápání bloku byl vždy stejný. Přesně takovou měl představu o tom, jak by mohl fungovat vývoj front-endu. [28]

## 4.2 Utility X inline styly

Tento přístup může na první pohled působit stejně jako styly vložené. Vypisujeme třídy, které mají konkrétní hodnotu a vlastnost. Je tam však podstatný rozdíl.

V případe inline stylů můžeme u jednoho elementu napsat velikost písma 14px, u jiného 13px a nemáme víceméně žádná omezení nebo pravidla. Pokud si však napíšeme systém pomocí utility CSS, předem si definujeme přesný počet barev, přesné velikosti písma nebo konkrétní velikosti vnitřních a vnějších okrajů. Najednou nevybíráme náhodně ze všech možných hodnot, ale pouze z několika předem definovaných, které si stanovíme nebo máme v zadání.

[26][29]

Následující rozdíl je v dnešní době velice důležitý, kdy při využití vložených stylů nelze využít media queries, které se starají o změnu chování na jiné velikosti rozlišení. [30]

Další velice důležitou odlišností jsou pseudotřídy. U vložených stylů nemáme možnost využít například :hover nebo :focus, které nám pomáhají zvýraznit styl prvků, na kterých máme ukazatel myši nebo kurzor. [30]

V případě, že potřebujeme změnit hodnotu, postačí nám nalézt správnou CSS třídu a u té ji změnit. Nemusíme měnit hodnotu u každého elementu separátně, což je další nevýhoda, kterou inline styly mají oproti utilitám. [26][29]

## 4.3 Utility a komponenty

Již v úvodu byla komponenta postavená čistě na utilitách ukázána, určitě to tedy jde. Jak je to se složitějšími responzivními komponentami? Ukažme si příklad:

```
1  <div class="max-w-md mx-auto bg-white rounded-xl shadow-md
       overflow-hidden md:max-w-2xl">
2    <div class="md:flex">
3      <div class="md:flex-shrink-0">
4        <img class="h-48 w-full object-cover md:w-48" src="/img/store
             .jpg" alt="Man looking at item at a store">
5      </div>
6      <div class="p-8">
7        <div class="uppercase tracking-wide text-sm text-indigo-500
             font-semibold">Case study</div>
8        <a href="#" class="block mt-1 text-lg leading-tight
             font-medium text-black hover:underline">Finding customers
              for your new business</a>
9        <p class="mt-2 text-gray-500">Getting a new business off the
```

```
        ground is a lot of hard work. Here are five ideas you can
         use to find your first customers.</p>
    </div>
  </div>
</div>
```

Příklad 5: Složitější komponenta pomocí utilit

Pokud bychom takovýchto komponent používali několik na různých místech, stane se pro změnu těžko spravovatelné HTML. Jak to tedy dát dohromady? [31]

Jednou z možností je využití systémů, které umožňují vytvářet šablony nebo komponenty (React, Vue, Latte a jiné), kde si předem definujeme strukturu. Následně tyto předdefinované složky využíváme a v případě změny se projeví ve všech místech, kde byla šablona nebo komponenta využita. Tento přístup samozřejmě nemusí být zcela vyhovující. Ať už z důvodu, že se HTML zdá stále nepřehledné nebo kvůli samotnému systému šablon/komponent. [31]

Druhá cesta jsou takzvané BEM komponenty, které využívají utility pro často opakované prvky nebo takové, které by mohly zbytečně tvořit modifikátory (margin, padding a jiné). Zpřehlední se díky tomuto přístupu HTML a CSS zkrátíme o často opakované prvky. [31]

## 4.4 Výhody a nevýhody

Většina z výhod a nevýhod tohoto přístupu mohla vyplynout již z předchozích příkladů. Rozhodl jsem se je tedy shrnout v následných podkapitolách.

### 4.4.1 Výhody

Jednou z hlavních výhod je snadné psaní nových komponent. Vychází to z toho, že CSS třídy jsou snadno zapamatovatelné a zároveň většinu CSS, které pro vytvoření komponenty potřebujeme, v CSS nejspíše už máme. Můžeme se tedy

soustředit čistě na psaní HTML a nemusíme plýtvat pozornost na přepínání mezi CSS a HTML. [26]

Dosáhneme velmi nízké velikosti CSS souboru. V CSS budeme mít definované pouze určité vlastnosti a hodnoty, které potřebujeme a opakovaně využíváme. Tím pádem brzy budeme mít veškeré potřebné styly pro projekt vytvořené a nebudeme muset přidávat nové CSS třídy. [26]

Vzhledem k tomu, že třídy obsahují vlastnost a hodnotu, nemají žádný kontext a můžeme je využít na jakýkoliv element, který může danou vlastnost využívat. Třída .flex vždy nastaví prvku display vlastnost na hodnotu flex, nehledě na to, zda element bude sekce, článek nebo div. [26]

### 4.4.2 Nevýhody

V případě, že se CSS skládá z tříd, které mají pouze jeden význam, bude složité z nich poskládat přehledné HTML, pokud budeme tvořit složitější komponentu. Získáme tedy obsahově krátké CSS, ale zaplatíme za to velmi špatně čitelným HTML u komplikovaných prvků. [26]

Další kámen úrazu může být HTML v případě, že nevyužíváme šablonovací systémy nebo komponenty. Pokud se rozhodneme něco upravit, musíme to upravit na všech místech, kde jsme třídu využili a potřebujeme ji změnit. V případě „najít a nahradit“ bychom mohli třídu změnit nebo odebrat i u jiného prvku, u kterého to provést nechceme. [26]

Pokud chceme zajistit responzivní chování, musíme u elementu pro každý breakpoint a vlastnost přidat další CSS třídu (sm:třída, md:třída a jiné), abychom upravili hodnotu dané vlastnosti. Stejně je tomu u pseudotříd, kde pro každou musíme definovat samostatnou třídu (hover:třída, focus:třída a jiné). Tento proces nepůsobí úplně příznivě. V případě pseudoelementů to není prakticky možné, jelikož ty jsou mimo princip utility tříd. [26]

# 5 JavaScript

JavaScript je interpretovaný nebo také just-in-time kompilovaný programovací jazyk, který vynalezl Brendan Eich v roce 1995 a od roku 1997 je standardem ECMA. Nejčastěji bývá označován jako skriptovací jazyk pro webové stránky, můžeme jej však využít i v prostředí mimo prohlížeč. Například pro serverovou část pomocí Node.js. Pokud jej využijeme jako skript pro webové stránky, odesílá se skript se stránkou do prohlížeče klienta a tam je následně spuštěn. Oproti tomu v případě Node.js, jako serverového skriptu, se veškerá logika vykoná na serveru a klient obdrží pouze odpověď s daty. [32][33]

Slabé stránky, které tento jazyk v prohlížeči má, jsou například možnost zakázat JavaScript v prohlížeči nebo nemožnost práce se soubory či systémovými objekty. Další podstatná věc, která může být částečně výhoda i nevýhoda je ta, že zaslané JavaScriptové soubory jsou dostupné z prohlížeče a můžeme si je zobrazit. Lze si tedy nastudovat skripty použité u jiné webové stránky a získat tak nové vědomosti. Musíme si však dát pozor v případě soukromých údajů a neudávat je, jelikož i ty by bylo možné zobrazit. [32][33]

## 5.1 ES6

ES6, označovaný také jako ECMAScript 2015, je šestá edice ECMAScript jazykové specifikace. Od publikace první edice se ECMAScript stal jedním z nejpoužívanějších univerzálních jazyků. Nejvíce je znám jako jazyk zabudovaný do webových prohlížečů, ale byl také široce užíván pro servery a vestavěné aplikace. Tato edice je nejrozsáhlejší aktualizací ECMAScriptu od prvního vydání v roce 1997. [34]

ECMAScript samotný je založen na několika původních technologiích. Nejznámější z nich jsou JavaScript (Netscape) a JScript (Microsoft). Jazyk vynalezl Brendan Eich v Netscapu a poprvé se objevil v prohlížeči Navigator 2.0. Objevil se také ve všech následujících prohlížečích od Netscapu a Microsoftu, počínaje Internetem Explorer 3.0. [34]

Cílený vývoj začal roku 2009 a cílem šesté edice bylo umožnit lepší podporu pro velké aplikace a vytváření knihoven. Některé hlavní vylepšení jsou moduly, deklarace tříd, iterátory a generátory, sliby pro asynchronní programování (promises) a vzory ničení (destructuring). Knihovna integrovaných modulů byla rozšířena o podporu další abstrakce dat, a to včetně map, sad, polí binárních numerických hodnot a s tím další podpora doplňkových znaků Unicode v řetězcích a regulárních výrazech. [34]

## 5.2 ECMAScript 2016 a 2017

Následující verze ECMAScriptu již nenesou označení ESX (kde X zastupuje číslo verze), ale jsou pojmenovány dle roku. Další roky po ES6 tedy vznikají ECMAScript 2016 a ECMAScript 2017.

Verze 2016 přináší novinku exponenciální operátor **, který funguje stejně jako Math.pow(x,y). Exponenciální přiřazení **=, které umocní proměnnou číslem za rovná se a novou hodnotu do proměnné uloží. Poslední funkcí je includes, která slouží k nalezení prvku v poli a dle výskytu vrací true nebo false. [35]

Edice 2017 přináší také zajímavé novinky. První z nich je String padding, který nabízí 2 metody padStart a padEnd. První parametr metod určuje délku výsledného řetězce a druhý znak nebo slova, která se připojí na konec či začátek původního řetězce. Následující metody jsou pro objekty a jedná se o metodu Object entries a Object values. Metody přijímají jako parametr objekt a podle zvolené metody nám navrátí klíče nebo hodnoty, které jsou pod těmito klíči uložené. Poslední jsou asynchronní funkce, které se mohou hodit, pokud zasíláme například dotaz na Rest API a očekáváme odpověď, která však není instantní, a my na ní potřebujeme vyčkat před spuštěním navazujícího kódu. Jedná se o alternativu funkce then. [35]

# 6 Knihovna

Knihovna poskytuje sadu pomocných funkcí, objektů nebo modulů, které voláme z aplikace pro konkrétní funkce. Obvykle se zaměřuje na úzký rozsah (např. práce s řetězci, vstupně-výstupní operace a jiné), takže jejich aplikační rozhraní (API), které služby poskytují bývají menší a vyžadují méně závislostí. Důvod, proč jsou knihovny užitečné je ten, že můžeme využít kód, který už napsal a vymyslel jiný vývojář, nebo který jsme si napsali již dříve. Nemusíme tedy znovu vynalézat kolo v případě, že nám knihovna vyhovuje. [36]

## 6.1 React

React je deklarativní, efektivní a flexibilní JavaScriptová knihovna pro vytváření uživatelských rozhraní. Umožňuje skládat složité uživatelské rozhraní z malých, izolovaných a znovupoužitelných částí kódu, které se nazývají komponenty. Můžeme jej využít v míře, jakou uznáme za vhodnou. Například pro interaktivitu u jednoduché HTML stránky nebo klidně komplexní aplikaci. [37][38]

Možnosti použití:

- CDN linky - Nejčastěji používané do již existujících projektů.
- Create React App - Nová aplikace, která vytvoří vývojářské prostředí. Toto prostředí umožňuje využívat nejnovější featury JavaScriptu a optimalizuje aplikaci pro produkci.

Create React App však vyžaduje nainstalovaný Node ve verzi 8.10 nebo novější a npm ve verzi 5.6 nebo novější na našem zařízení. [39][40]

### 6.1.1 JSX

JSX je rozšíření syntaxe k JavaScriptu a zápisem je skoro identické s HTML. Není vyžadováno jej využívat v kombinaci s Reactem, ale je to považováno za

vizuální pomůcku při práci s UI uvnitř JavaScriptového kódu. Umožňuje také Reactu zobrazovat užitečné chybové a varovné zprávy. [41]

Jednoduchá ukázka JSX:

```
1 <div>
2   <h1 className="title">Hello, world!</h1>
3   <h2>My name is {person.name}.</h2>
4 </div>
```

Příklad 6: Ukázka JSX - React

Jak bylo zmíněno, příklad výše může působit jako HTML. Jde ovšem o syntaxi JSX, která nám do složených závorek umožňuje zapsat jakoukoliv validní syntaxi JavaScriptu. V případě atributu u elementu využíváme camelCase, jelikož JSX má blíže k JavaScriptu než k HTML. className tedy zastupuje atribut class, třeba tabindex bude tabIndex. className z toho důvodu, že class je v JavaScriptu rezervováno pro třídu. [41]

JSX je kompilován pomocí Babelu do funkce React.createElement(). Tato funkce provádí několik kontrol, které nám pomohou psát bezchybný kód a vytvoří objekt, který se nazývá React element. Tyto elementy si lze představit jako popis toho, co chceme vidět na obrazovce. React tyto objekty čte a využívá je ke konstrukci modelu DOM a jeho aktualizaci. [41]

### 6.1.2 Vykreslování elementů

Narozdíl od DOM elementů jsou React elementy obyčejné objekty. O aktualizaci DOM se stará React DOM, který zajistí, aby odpovídal stavu React elementům. [42]

Pokud bychom chtěli vykreslit element do DOMu, budeme potřebovat například div, který si označíme unikátním identifikátorem, ideálně root. Nazývá se root (kořenový) DOM uzel, protože vše uvnitř něj bude spravováno pomocí React DOM. Aplikace, které jsou postavené na Reactu obvykle mají jediný root

element. Pokud však integrujeme React do existující aplikace, můžeme mít izolovaných root DOM uzlů kolik potřebujeme. Pro vykreslení React elementu do kořenové DOM uzlu využijeme ReactDOM.render(), kde je oba předáme jako parametry.

Elementy jsou neměnné, jakmile vytvoříme prvek, nemůžeme změnit jeho podřízené prvky ani atributy. V tomto případě je jediný způsob, jak aktualizovat uživatelské rozhraní a tím je vytvořit nový prvek a předat jej znovu do metody ReactDOM.render(). [42]

V případě takovéto úpravy React DOM porovná element a jeho potomky s předchozím a aktualizuje pouze části v DOM, které se změnili. [42]

### 6.1.3 Komponenta

Koncepčně jsou komponenty podobné funkcím JavaScriptu. Přijímají libovolné vstupy (nazývané props) a vrací React elementy popisující, co by se mělo zobrazit na obrazovce. Pro definování komponenty můžeme využít JavaScriptovou funkci nebo ES6 třídu. [43]

Komponenty v JSX zapisujeme jako tagy podobné HTML a vždy musí začínat velkým písmenem. Pokud React vidí element, který reprezentuje komponentu definovanou uživatelem, předá do ní JSX atributy a potomky jako jeden objekt. Tento objekt jsou zmiňované props a jsou Read-Only což znamená, že nemohou být nikdy upraveny. [43]

### 6.1.4 Stav a životní cyklus

Stav v Reactu je podobný vstupům (props), ale je soukromý a plně kontrolovaný komponentou. Zásadní je to, že pokud se stav komponenty změní správně pomocí metody setState, komponenta se automaticky znovu vykreslí, aby odpovídala aktuálnímu stavu. Zbavíme se tedy nutnosti volat dokola metodu ReactDOM.render(). [44]

První vykreslení komponenty v modelu DOM se nazývá „montáž“ (moun-

ting) a smazání z modelu DOM „odpojení“ (unmounting). React nám umožňuje deklarovat speciální metody v komponentě, která je vytvořena jako třída (u funkcionální komponenty tyto metody nejsou, ale lze to řešit jinak), abychom mohli spustit nějaký kód, když se komponenta připojí nebo odpojí. Díky tomu můžeme například uvolnit zdroje, které komponenta čerpala nebo provést dotaz na API, které nám poskytne potřebné data pro komponentu. [44]

### 6.1.5 Zpracování událostí

Zpracování událostí u React elementů je podobné zpracování událostí u DOM elementů. Obsahuje několik rozdílu v syntaxi:

- React události jsou pojmenovány pomocí camelCase více než lowercase
- Pomocí JSX předáme funkci jako obsluhu události (eventHandler) spíše než text

Další odlišností je, že nemůžeme vrátit false, abychom zabránili výchozímu chování (prevent default) v Reactu. Musíme preventDefault zavolat explicitně. Navíc nemusíme obecně volat addEventListener, abychom vytvořili posluchače. Místo toho stačí poskytnout posluchače, když je prvek zpočátku vykreslen. [45]

### 6.1.6 Podmíněné vykreslování

Podmíněné vykreslování funguje stejně, jako podmínky v JavaScriptu. Můžeme využít operáty jako if nebo podmíněný operátor k vytvoření elementu, který představuje aktuální stav. [46]

V podstatě nám to umožňuje na základě stavu nebo pevně dané podmínky vykreslit různé komponenty. Například, pokud budeme u blogu vykreslovat příspěvky, můžeme ověřit zda nějaké příspěvky máme k dispozici. Na základě toho buď vykreslíme komponentu pro příspěvky nebo text o nenalezení příspěvku. [46]

# 7 Tailwind

Tailwind je utility-first CSS open source framework vytvořený Adamem Wathanem, který měl své první vydání 1. listopadu 2017 s označením v0.1.0. Původně se jednalo o vedlejší projekt, který však Adama natolik nadchnul, že se rozhodl ve vývoji pokračovat. Klíčovým bodem byla úspěšná kniha Refactoring UI z prosince roku 2018, kterou vydal se Stevem Schogerem. Díky zmíněné knize měl velký rozpočet a věděl, že existují způsoby, jak vybudovat komerční nabídku okolo frameworku samotného. Rozhodl se tedy využít získané znalosti a 13. května 2019 přišlo vydání verze v1.0, která signalizovala závazek ke stabilitě a posun hranic novými funkcemi v menších vydáních. Těchto menších vydání bylo celkem 9 a přinesly nové funkce jako stylování placeholderu, CSS grid, přechody, transformace, animace, nástroje pro layout, gradienty a spoustu dalších. [47][48][49]*

18. listopadu 2020 následovala první významná aktualizace, kterou přinesla verze v2.0. Tato aktualizace obsahovala spoustu zajímavých novinek. Objevila se zcela nová paleta barev s 220 barvami (22 barev s 10 odstíny), podpora tmavého režimu, nový 2XL breakpoint pro 1536px a výše, nové utility pro zaoblené gradienty, utility-friendly styly pro formulář, výchozí výška řádku, direktivita @apply pro jakoukoliv třídu (hover, focus a jiné), nové nástroje pro přetékání textu, rozšíření variant a několik dalších. Problém nastane pro uživatele Interneru Explorerer včetně verze 11, jelikož s ním není tato aktualizace kompatibilní. Pokud tuto podporu potřebuje, lze využít verzi v1.9, která je plně kompatibilní. U existujících projektů při přechodu ze starších verzí byla navržena tak, aby nebylo nutné upravovat příliš mnoho věcí a dokumentace nabízí podrobnosti k migraci na tuto novou verzi. [49]

## 7.1 Tailwind UI

Po vydání Tailwindu verze v1.0 vymýšleli Wathan a Schoger, co vlastně bude byznys plán pro Tailwind. Zkusili spoustu prototypů a nápadů, ale nakonec zů-

stali u pokračování dnešního Tailwind UI. Myšlenka byla taková, že by tvůrci a vývojáři frameworku vytvářeli responzivní profesionální komponenty, které by si komunita mohla zkopírovat a využívat ve svých projektech. První ztvárnění této myšlenky Wathan zveřejnil v březnu 2019. Nápad vzbudil zájem, proto v únoru 2020 vyšel předčasný přístup. [48]

Aktuálně stránky Tailwind UI v rámci předčasného přístupu nabízí 3 balíčky, které si můžeme zakoupit:

- Application UI - získáme doživotní přístup ke všem UI komponentám na neomezené množství projektů, které mají aktualizace zdarma a přístup do uzavřené komunity.
- Marketing - získáme doživotní přístup ke všem marketingovým komponentám na neomezené množství projektů, které mají aktualizace zdarma a přístup do uzavřené komunity.
- Application UI + Marketing - balíček, který je kombinací výše zmíněných a nabízí výhodnější cenu, než kdybychom balíčky zakoupili odděleně.

Zakoupením balíčku získáme přístup k více než 300 komponentám a slib nových komponent každý měsíc, dokud tvůrci budou mít nápady. [50]

## 7.2 Instalace

Tailwind nabízí samostatné návody pro instalaci v kombinaci s nástroji Vue.js, React.js, Lavarel, Gatsby, Nuxt.js a Next.js. Pokud však instalaci chceme využít pro projekt, kde nevyužíváme ani jednu z těchto technologií, máme na výběr ze 3 možností: [51]

- Instalace jako PostCSS plugin - všeobecně doporučená metoda
- Instalace bez využití PostCSS - doporučená metoda pro malé projekty nebo vyzkoušení frameworku

- Použití pomocí CDN linku - nedoporučovaná metoda

Pokud se rozhodneme využít nejjednodušší způsob v podobě CDN linku, narazíme na několik omezení: [51]

- Nebudeme mít možnost upravovat výchozí šablonu Tailwindu.
- Přijdeme o možnost využívat direktivy jako @apply a @variants.
- Nelze si povolit dodatečné varianty jako group-focus nebo group-hover.
- Nemůžeme si nainstalovat pluginy třetích stran.
- Přijdeme o možnost zbavit se nevyužitích stylů.

## 7.3 Základní koncepty

Tailwind je utility-first CSS framework, což znamená, že je celý postaven na třídách, které mají jen jeden účel. Rozhodl jsem se tedy představit jeho pojetí důležitých prvků, jako je například responzivita, pseudotřídy nebo tvorba komponent.

### 7.3.1 Responzivní vzhled

Každou utility třídu, kterou framework nabízí, můžeme podmínit breakpointem a tím změnit hodnotu vlastnosti při různě velkém rozlišení. Ve výchozím nastavení máme k dispozici 5 breakpointů, které pomocí konfiguračního souboru můžeme změnit: [52]

- sm - 640px a výše
- md - 768px a výše
- lg - 1024px a výše
- xl - 1280px a výše

- 2xl - 1536px a výše

Breakpoint na utility třídu snadno aplikujeme tím, že před ní přidáme název breakpointu a dvojtečku: [52]

```
1 <img class="w-16 md:w-32 lg:w-48" src="...">
```

Příklad 7: Ukázka responzivity - Tailwind

Tento systém je ve výchozím nastavení navržen pro mobile first. Znamená to tedy, že pokud breakpoint zvolíme, aplikuje se na zvolené a vyšší rozlišení. Viz ukázka výše, kde výchozí šířku určuje třída w-16, od rozlišení 768px a výše je nahrazena třídou w-32 a při 1024px nebo výše ji nahrazuje třída w-48. [52]

Pokud bychom chtěli cílit pouze na jeden breakpoint, musíme přidat pravidlo pro o 1 vyšší breakpoint, abychom opět změnili chování: [52]

```
1 <div class="bg-teal-500 lg:bg-red-500 xl:bg-teal-500">
2 </div>
```

Příklad 8: Ukázka pro samostatný breakpoint - Tailwind

### 7.3.2 Pseudotřídy a jiné varianty

Pseudotřídy v Tailwindu náleží pod takzvané varianty, kam spadá i třeba dark mode nebo výše zmíněné varianty pro responzivitu (sm, md a jiné). Použití je tedy stejné jako v případě responzivního designu. Varianty jsou také responzivní a lze je tedy kombinovat s responzivním prefixem. Responzivní prefix musí být vždy první, v opačném pořadí toto spojení nefunguje: [53]

```
1 <h2 class="text-2xl text-red-400 lg:hover:text-black">Spravne</h2>
2
3 <h2 class="text-2xl text-red-400 hover:lg:text-black">Chybne</h2>
```

Příklad 9: Kombinace pseudotřídy a breakpointu - Tailwind

Zároveň ve výchozím nastavení nejsou pro všechny utility povoleny všechny varianty z důvodu velikosti souboru, ale jen nejčastěji užívané kombinace. Toto nastavení lze snadno v případě potřeby upravit v konfiguračním souboru Tailwindu. [53]

Pokud bychom potřebovali varianty, které Tailwind nepodporuje, jako například required, lze je přidat pomocí vlastního pluginu v konfiguračním souboru. [53]

### 7.3.3 Pseudoelementy

Varianty pro pseudoelementy bohužel defaultně v Tailwindu nenalezneme. Můžeme pro to však využít například plugin tailwindcss-pseudo-elements. Tento plugin za nás varianty pro ::before, ::after a další vytvoří. Následné využití je podobné jako v případě breakpointů a pseudotříd.

Snazší a přehlednější varianta dle mého názoru je ta, že pro toto využijeme klasický přistup pomocí selektor::pseudoelement a CSS v kombinaci s direktivou @apply, o které se více zmíním u komponent a direktiv jako takových.

### 7.3.4 Komponenty

Pro vytvoření komponenty nabízí Tailwind 3 tipy. První z nich je využití částečné šablony nebo Javascriptové komponenty. Tento přístup snadno využijeme pomocí React komponent, Blade komponent, Twig includes a podobných. [54]

Pro tvorbu malých komponent, jako jsou tlačítka nebo prvky formuláře, může být tvorba šablon nebo komponent zbytečně náročná oproti jednoduché CSS třídě. V takovém případě lze využít direktivu @apply, kterou bych přirovnal k mixinu z vytvořených tříd, existujícímu v Less preprocesoru. Zjednodušeně řečeno nám umožňuje využívat styly jiné CSS třídy. Pokud tedy nějakou utility třídu změníme, změny se projeví i do vytvořené komponenty.

[54]

Poslední variantou je napsání pluginu pro komponentu. Tento přístup je doporučeno využít v případě, že chceme publikovat naše Tailwind komponenty jako knihovnu nebo zjednodušit sdílení napříč více projekty. [54]

### 7.3.5 Direktivy a funkce

Tailwind si pro nás připravil 6 zajímavých direktiv: [55]

- @tailwind - Umožňuje vložit do našeho CSS třídy pro základ, komponenty, utility a responzivní varianty, které byly zaregistrované pomocí pluginů nebo patří Tailwindu.
- @apply - Slouží k vložení existujících tříd do našich vlastních CSS. Třídy se u této direktivy používají na základě pořadí v původním CSS, což může ovlivnit chování.
- @layer - Pomocí této direktivy řekneme Tailwindu kam umístit námi vytvořenou třídu (základ, komponenty nebo utility). Tailwind následně dle toho automaticky umístí jakékoliv CSS jako odpovídající @tailwind pravidlo. Nemusíme se tedy starat o problém se specifičností způsobenou pořadím.
- @variants - Umožňuje nám generovat responzivní, hover, focus nebo jiné varianty pro námi vytvořené utility. V tomto případě se varianty generují v pořadí, v jakém je zadáme. Záleží tedy, jaké variantě chceme dát větší prioritu.
- @responsive - Tato direktiva je ve své podstatě zkrácená verze pro @variants responsive. Slouží tedy čistě pro vygenerování responzivních variant pro námi vytvořenou utilitu.
- @screen - Tuto direktivu můžeme využít místo deklarace media query a umožňuje nám odkazovat na breakpointy, které již máme nastavené

podle názvu. Například místo @media (min-width: 640px){} můžeme využít @screen sm{}.

V případě funkcí nabízí pouze 1 a nazývá se theme. Tato funkce nám umožňuje přístup k hodnotám v konfiguraci pomocí tečkové notace. [55]

Lze to využít jako užitečnou alternativu k @apply, pokud chceme odkazovat na hodnotu z konfigurace pro část deklarace: [55]

```
.content-area {
  height: calc(100vh - theme('spacing.12'));
}
```

Příklad 10: metoda theme, odkaz na celočíselnou hodnotu - Tailwind [55]

Pokud potřebujeme získat přístup k hodnotě, která není celé číslo, lze využít notaci hranatých závorek: [55]

```
.content-area {
  height: calc(100vh - theme('spacing[2.5]'));
}
```

Příklad 11: metoda theme, hodnota není celé číslo - Tailwind [55]

Tailwind využívá syntaxi vnořeného objektu pro definování základní palety barev, pokud tedy chceme získat přístup k hodnotě vnořené barvy, musíme také využít tečkovou notaci: [55]

```
.btn-blue {
  background-color: theme('colors.blue.500');
}
```

Příklad 12: metoda theme, vnořená hodnota barvy - Tailwind [55]

## 7.4 Přizpůsobení

Tailwind je framework pro vytváření uživatelských rozhraní na míru, je proto od základu navržen s ohledem na přizpůsobení. Ve výchozím nastavení vyhledá Tailwind volitelný soubor tailwind.config.js v kořenovém adresáři projektu, kde lze definovat libovolná přizpůsobení. Každá část nastavení je volitelná a ty, které nenadefinujeme, využijí výchozí konfiguraci.[56]

Konfigurační soubor snadno vytvoříme pomocí Tailwind CLI, které je součástí instalace npm balíčku tailwindcss:[56]

```
npx tailwindcss init

// obsah vytvoreneho souboru tailwind.config.js
module.exports = {
  purge: [],
  darkMode: false, // or 'media' or 'class'
  theme: {
    extend: {},
  },
  variants: {
    extend: {},
  },
  plugins: [],
}
```

Příklad 13: Vytvoření konfiguračního souboru - Tailwind

Tento soubor můžeme vygenerovat i s vlastním názvem, který přidáme za příkaz init: [56]

```
npx tailwindcss init tailwindcss-config.js
```

Příklad 14: Vytvoření konfiguračního souboru s vlastním názvem - Tailwind

Pokud používáme vlastní název souboru, je nutné jej zadat do konfigurace PostCSS, když zahrnujeme Tailwind jako plugin: [56]

```
// postcss.config.js
module.exports = {
  plugins: {
    tailwindcss: { config: './tailwindcss-config.js' },
  },
}
```

Příklad 15: specifikace konfiguračního souboru v PostCSS - Tailwind

Pokud konfigurační soubor pro PostCSS nemáme, můžeme jej vytvořit ručně nebo pomocí příkazu: [56]

```
npx tailwindcss init -p

// obsah vytvoreneho souboru
module.exports = {
  plugins: {
    tailwindcss: {},
    autoprefixer: {},
  },
}
```

Příklad 16: Vytvoření konfiguračního souboru PostCSS - Tailwind

Všechny tyto metody vygenerují v podstatě prázdnou konfiguraci, kde můžeme výchozí nastavení upravit nebo nahradit a udržet ji obsahově velmi malou. [56]

V případě, že bychom měli zájem vygenerovat konfigurační soubor, který v sobě obsahuje všechny základní nastavení Tailwindu, lze toho docílit pomocí příkazu: [56]

```
1 npx tailwindcss init --full
```

Příklad 17: Vytvoření konfiguračního souboru PostCSS - Tailwind

Tento způsob však není doporučen, jelikož konfigurační soubor by byl zbytečně velký. [56]

### 7.4.1 Šablona

Sekce šablony (theme) v konfiguračním souboru slouží pro definování palety barev, fontů, breakpointů a dalších. Sem tedy náleží vše, co je spojeno s vizuálním designem stránky. [56]

### 7.4.2 Varianty

Sekce pro varianty (variants) nám umožňuje kontrolovat, jaké varianty jsou vygenerované pro jednotlivé základní pluginy. Zde můžeme povolit například pseudotřídy, responzivitu a veškeré námi nebo Tailwindem nadefinované varianty. [56]

### 7.4.3 Pluginy

Plugin sekce (plugins) nám povoluje registrovat pluginy, které mohou být použity pro generování extra utilit, komponent, základních stylů nebo vlastních variant. Zde si můžeme přidat například náš vlastní plugin, který nám vytvoří potřebné komponenty. [56]

### 7.4.4 Předvolby

Sekce předvolby (presets) nám umožňuje změnit základní konfiguraci Tailwindu na vlastní. Dovoluje nám v podstatě nastavit ostatní sekce, jako jsou například pluginy, varianty nebo šablona, které se stanou základním nastavením místo nastavení Tailwindu. [56]

## 7.5 Tailwind a Tachyons

Tachyons je framework, který je stejně jako Tailwind založený na utility třídách. Dokonce jej tvůrce Tailwindu v roce 2017 označil za skvělý projekt. V určitém ohledu bychom mohli označit Tailwind za následníka Tachyonu. Rozhodl jsem se tedy mezi nimi porovnat několik věcí: [29]

- Přizpůsobení CSS
- CSS třídy
- Velikost
- Komponenty a jejich tvorba

### Přizpůsobení CSS

V případě Tailwindu nalezneme v dokumentaci popsané všechny části konfiguračního souboru, včetně vygenerování konfiguračního souboru a ukázky konkrétního nastavení. Můžeme si tak snadno rozšířit nebo zcela přepsat základní strukturu CSS. Celkově v ohledu přizpůsobení nabízí Tailwind daleko více možností. [56]

U Tachyons tento zásah do struktury lze provést při stažení kompletního CSS souboru nebo zasláním požadavku na jejich API, s konfiguračním souborem, které nám CSS vygeneruje včetně dokumentace. [57]

### CSS třídy

Pokud si otevřeme CSS soubory obou frameworků, budou se lišit nejen počtem tříd, ale také způsobem pojmenování jednotlivých utilit.

Tachyons nabízí ve výchozím nastavení něco přes 2000 CSS pravidel. V případě Tailwindu jich máme k dispozici více než 14 000. Tailwind tedy v základu nabízí daleko větší možnost výběru a rozhodování. [58]

Zmínil jsem se také o pojmenování tříd, které mi osobně více vyhovuje u Tailwindu. K porovnání využiji ukázku velikostí písma. U Tachyons jsou

třídy pro velikost písma pojmenované fx, kde x je číslo označující velikost (1-6). Tailwind si vybral styl zápisu text-x, kde x nahrazuje název velikosti (xs, sm, base a jiné). Líbí se mi využití klíčového slova text, kdy je na první pohled vidět, co konkrétně bude třída ovlivňovat. [47][57]

### Velikost CSS

Velikost se odvíjí od předchozího bodu, kdy Tachyons má výrazně méně tříd a je tedy o dost menší. V základním nastavení má velikost 13kb po gzipu, kterou lze ještě snížit využitím jen potřebných modulů. [57]

Tailwind oproti tomu má v nové verzi po gzipu 294kb. V případě rozšiřování o vlastní utility velikost vcelku naroste, z důvodu generování všech variant, které povolíme v konfiguraci. Velikost nám však pomůže snížit nástroj pro produkci PurgeCSS, který se stará o odebrání nevyužitých CSS. Další možnost je odebrat základní pluginy, které nevyužíváme. Tailwind na svém webu uvádí, že spoustu projektů po této úpravě nepřesáhne 10kb. [47][59]

### Komponenty a jejich tvorba

Komponenty jsou základní prvek, který tvoříme prakticky u jakéhokoliv projektu. Ať už v podobě tlačítka nebo něčeho složitějšího.

Tachyons pro tvorbu komponent doporučuje využití šablonovacích systémů nebo komponent, které nabízí Vue.js, React.js a podobné nástroje. [29]

Podobné doporučení nalezneme i u Tailwindu, který však nabízí ještě jednu zajímavější metodu. Pomocí direktivy @apply v CSS můžeme pro sestavení komponenty využít již definované utility třídy. Přesune se nám tedy abstrakce do CSS a zpřehledníme tím HTML. Zároveň nemusíme napsat žádné CSS, protože utility z kterých komponentu skládáme, již v CSS máme. [54]

Pokud se podíváme na komponenty, které poskytují přímo frameworky, tak u Tachyons jich nalezneme okolo 150, které jsou zdarma. V případě Tailwindu nabízí skrze Tailwind UI více než 300, které jsou však placené. [50][57]

# 8 Tailwind a Bootstrap

## 8.1 Bootstrap

Bootstrap, původně vytvořen designérem a vývojářem v Twitteru, se stal jedním z nejpopulárnějších front-end frameworků a open source projektů na světě. Umožňuje nám rychle navrhnout a upravit mobile-first stránky. [60]

Při využívání Bootstrapu si můžeme stránku pomyslně rozdělit do 12 sloupců, ve kterých lze ovlivnit rozložení pomocí kontejnerů, řádků a sloupců. Tento systém podporuje 6 breakpointů pro responzivitu, které jsou založené na minimální šířce. [60]

Stejně jako Tailwind, nabízí Bootstrap možnost upravit si části frameworku podle vlastních představ. Můžeme si tedy například vytvořit vlastní komponentu, upravit systém barev, hranice breakpointů a tak dále. [60]

Velmi zajímavou částí jsou zmíněné komponenty, které jsou základem tohoto frameworku. Bootstrap nabízí předem vytvořené komponenty, které nám velmi usnadní tvorbu webů a v mnoha případech není potřeba ani kreativita, jelikož máme k dispozici sadu komponent. Pro komponenty jako je menu, carousel a podobné, poskytuje framework i JavaScript. Nemusíme si tedy vymýšlet a psát vlastní. [60]

Oficiální stránky nabízí i zajímavé ukázky jako dashboard, stránku pro přihlášení nebo blog, které si lze zdarma stáhnout. Součást, kterou ocení i lidé, kteří se v kódování příliš neorientují, je možnost zakoupení kompletní šablony. [60]

## 8.2 Porovnání

### Základ

První základní rozdíl je koncept, na kterém jsou frameworky založeny. Tailwind prosazuje utility-first přístup, pomocí kterého následně vytváří komponenty. Naproti tomu Bootstrap využívá utility jen jako doplněk k předem vytvořeným

komponentám pomocí OOCSS. [47][60]

### Přehlednost HTML

V případě Tailwindu záleží na složitosti komponenty a konkrétním postupu, jakým komponenty tvoříme. Pokud například při složitější komponentě nevyužijeme direktivu @apply nebo plugin, přehlednost a orientace v HTML může být obtížnější:

```
1 <button class="px-4 py-1 text-sm text-purple-600 font-semibold
    rounded-full border border-purple-200 hover:text-white
    hover:bg-purple-600 hover:border-transparent focus:outline-none
     focus:ring-2 focus:ring-purple-600 focus:ring-offset-2">
    Message</button>
```

Příklad 18: Tailwind přehlednost HTML - Tailwind vs Bootstrap

Jedná se o stylizaci vcelku jednoduchého tlačítka. Už v tomto však případě můžeme vidět dlouhý seznam tříd, který pokud se v Tailwindu orientujeme, asi nebude tak velký problém. Představme si ovšem situaci, že by tlačítko bylo obaleno elementy, které mají podobný počet tříd.

Podobného výsledku bychom v Bootstrapu s pár úpravami CSS nebo utilitami, mohli dosáhnout přibližně takto: [60]

```
1 <button class="btn btn-outline-purple">Message</button>
```

Příklad 19: Bootstrap přehlednost HTML - Tailwind vs Bootstrap

Podobně by výsledek mohl vypadat s direktivou @apply u Tailwindu, kde bychom velikost přesunuli do CSS a vyšlo by to tedy obdobně, jako využít komponentu Bootstrapu. Rozdíl je v tom, že u Tailwindu komponentu opět seskládáme z předem daných utilit. [47]

### Velikost CSS

Základní velikost CSS Tailwindu, která není optimalizována pro produkci, je 294kb po gzipu. Velikost CSS po optimalizaci pro produkci se v případě Tailwindu velmi sníží a zároveň se odvíjí z předchozího prvku. Tedy zda se rozhodneme využít direktivu a rozšiřovat CSS o komponenty a nebo využijeme méně přehledné HTML, ale nebudeme nuceni přidávat moc nového CSS. [59]

V případě Bootstrapu je základní velikost po gzipu 23kb. Se základní strukturou Bootstrapu si však většinou nevystačíme, jelikož si budeme komponenty přizpůsobovat dle sebe nebo vytvářet nové a velikost nám více či méně naroste. [60]

Pro oba tyto frameworky můžeme využít nástroj, který jsem již zmiňoval a tím je PurgeCSS. Tento nástroj nám umožňuje zbavit se nevyužitých CSS. V případě Tailwindu jej nastavíme v konfiguračním souboru. Bootstrap pro tento prvek nemá oficiální příklad, ale odkazuje na stránky, kde se můžeme dozvědět více o využití tohoto nástroje. [59][60]

Výslednou produkční velikost tedy nelze jednoznačně porovnávat, jelikož se může odvíjet od více faktorů a bude záviset na tom, jaké zvolíme postupy u jednotlivých frameworků.

### Volnost návrhu

Základem Tailwindu jsou jednoúčelové třídy, které nemají kontext a dají se unikátně využít. Díky tomu máme víceméně neomezenou svobodu návrhu a prakticky vždy můžeme mít odlišný vzhled, který je nějakým způsobem originální. [47]

Na druhou stranu k tomu musíme mít určitou dávku kreativity. V tomto ohledu je snazší využít Bootstrap, který sice bez větších uprav přinese vždy podobný výsledek, nemusíme však nic moc vymýšlet a dokážeme vytvořit slušný web. [60]

### JavaScript pro komponenty

Bootstrap mimo CSS u komponent jako modal, carousel nebo dropdown poskytuje i funkční stránku v podobě JavaScriptu. To se může jevit jako pohodlná varianta oproti Tailwindu, který ke svým komponentám na Tailwind UI připravil komentáře, kde a jaké třídy pro funkčnost máme přidat. Musíme si však kód vytvořit vlastní. [50][60]

### Vhodné projekty

Pokud se podíváme na koncept Tailwindu, jedná se spíše o stavění na zelené louce. Pokud bychom byli tedy v situaci, kdy máme blížící se deadline, bylo by zbytečně časově náročné vše vytvářet od nuly. Mohli bychom si předem vytvořit komponenty pro tyto případy, jako tomu je u Bootstrapu. Tím bychom však ztratili jednu z hlavních podstat, kterou je unikátní vzhled. Naopak u dlouhodobějšího projektu Tailwind nabízí spoustu možností a svobodnou ruku při tvorbě libovolného vzhledu. [47]

Bootstrap naproti tomu nabízí sadu komponent, které můžeme hned používat, protože mají již stanovenou vzhledovou i funkční stránku. Nic nám tedy nebrání využít jej jak pro projekt, který má brzký deadline, tak pro dlouhodobý projekt. [60]

# 9 Praktická část

Praktickou částí bakalářské práce je tvorba ukázkových komponent, na kterých představuji některé z možností frameworku Tailwind. Hlavním prvkem je tvorba webu pro společnost Smart events pomocí tohoto frameworku a zajímavostí bude přidání techniky speech recognition na uživatelské formuláře. Výsledkem bude také porovnání klasického přístupu oproti tvorbě pomocí Tailwindu s ohledem na přehlednost HTML, efektivitu a časovou náročnost.

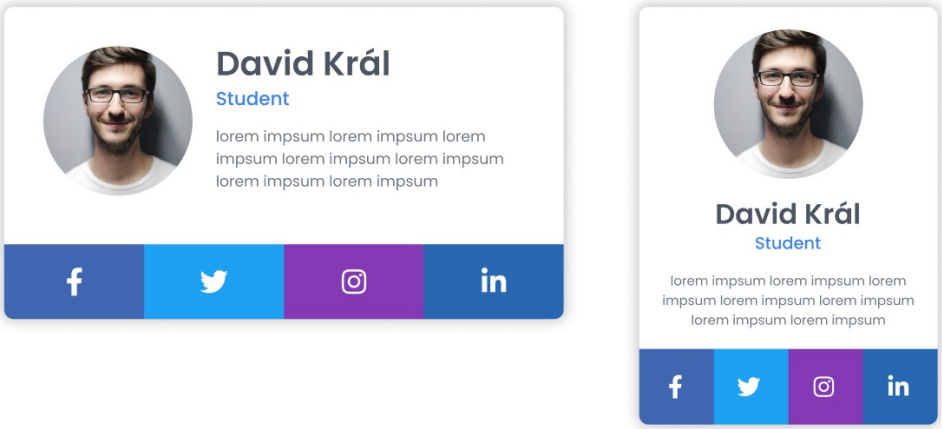
## 9.1 Ukázkové komponenty

Pro představení frameworku v praxi jsem si připravil tvorbu 2 komponent, které mi přijdou využitelné u většiny webů. U každé z ukázek se zaměřím na různé způsoby instalace, přizpůsobení a způsob tvorby komponenty v rámci frameworku. Součástí bude tvorba stejné komponenty pomocí klasického přístupu pro ukázku a porovnání zdrojového kódu.

### 9.1.1 Karta

První ukázková komponenta je karta. Často ji na webu uvidíme a můžeme využít k prezentaci náhledu příspěvku, produktu, služby nebo člena týmu. Typicky v kartě nalezneme náhledový obrázek, titulek, krátký popisný text, výzvu k akci (tlačítko, odkaz) a doplňující údaje (cena, počet zobrazení, hodnocení a jiné).

Pro tuto ukázku jsem si zvolil kartu prezentující osobu. Karta bude obsahovat náhledový obrázek, Jméno osoby, funkci, krátký popisek a spodní lištu se sociálními sítěmi. Samozřejmě bude responzivní, tedy bude využitelná jak na mobilních zařízeních, tabletech, tak desktopech.




Obrázek 1: Ukázka hotové karty - praktická část

### HTML

Část, která bude pro obě metody totožná, je rozložení HTML.

```html
<div class="card">
  <div class="card__top">
    <img class="card__image" src="./../src/img/image_small.jpg" alt="Náhledový obrázek - David Král" />
    <div class="card__info">
      <h2 class="card__title">David Král</h2>
      <h4 class="card__subtitle">Student</h4>
      <p class="card__description">lorem impsum lorem impsum lorem impsum lorem impsum lorem impsum lorem impsum lorem impsum </p>
    </div>
  </div>

  <div class="card__bottom">
    <ul class="card__links">
      <li class="card__link">
        <a href="#" target="_blank" title="odkaz na facebook">
          <img src="./../src/img/socialy/facebook-f-brands.svg" alt="logo facebook" />
        </a>
      </li>
      <li class="card__link">
        <a href="#" target="_blank" title="odkaz na twitter">
          <img src="./../src/img/socialy/twitter-brands.svg" alt="logo twitter" />
        </a>
      </li>
      <li class="card__link">
        <a href="#" target="_blank" title="odkaz na instagram">
          <img src="./../src/img/socialy/instagram-brands.svg" alt="logo instagram" />
        </a>
      </li>
      <li class="card__link">
        <a href="#" target="_blank" title="odkaz na linkedin">
          <img src="./../src/img/socialy/linkedin-in-brands.svg" alt="logo linkedin"
          />
        </a>
      </li>
    </ul>
  </div>
</div>
```

Obrázek 2: HTML karty - praktická část

Kartu jsem si rozdělil na horní a spodní část. Horní část se navíc ještě rozděluje na obrázek a informace. Spodní část obsahuje seznam odkazů, které jsou v tomto případě odkazy na sociální sítě.

Názvy tříd v tomto případě budou u Tailwindu sloužit pouze jako informační, aby se zlepšila přehlednost.

### Tailwind

U této ukázky jsem se rozhodl využít instalaci pomocí nejjednodušší varianty CDN linku. Stačí tedy do hlavičky webu umístit link tag, který nalezneme v dokumentaci Tailwindu.

V případě využití této varianty přijdeme o výhody, jako možnost upravení základní šablony, využití direktiv a podobné užitečné prvky. Utility třídy tedy budu aplikovat přímo do HTML.

Navíc budu potřebovat 1 utilitu pro stín a 4 utility pro pozadí sociálních sítí, jelikož potřebné barvy a stín v základní struktuře nejsou. V tomto případě nemáme k dispozici konfigurační soubor, je to tedy potřeba vyřešit vytvořením vlastního CSS souboru, který bude obsahovat potřebné styly:

```
1 .shadow-around {box-shadow: 0px 0px 16px 2px rgba(0, 0, 0, 0.25)}
2 .bg-fb { background: #4267b2 }
3 .bg-tw { background: #1da1f2 }
4 .bg-ig { background: #833ab4 }
5 .bg-in { background: #2867b2 }
```

Příklad 20: Potřebné utility, karta - Praktická část

### Klasický přístup

Pro klasický přístup si stanovím pár výchozích pravidel k dosažení úplné podobnosti:

```
1 * { margin: 0; padding: 0; box-sizing: border-box; font-family:
      inherit; }
2 body, html { font-family: "Poppins"; }
3 ul { list-style: none; }
```

Příklad 21: Výchozí pravidla, karta - Praktická část

### Postup tvorby

Pro klasický přístup potřebuji přidat stylování do CSS souboru. U Tailwindu mám připravené veškeré potřebné utility, tedy již budu doplňovat jen třídy v HTML. Aby byly komponenty totožné a nebylo nutné příliš sahat do CSS u Tailwindu, volil jsem hodnoty vlastností dle základní nabídky Tailwind utilit.

První krok bude upravení vzhledu obalu karty. V tomto případě je potřeba element omezit na maximální šířku, přidat zaoblené rohy a stín:

```
1 // klasicky pristup
2 .card { width: 91.666667%; max-width: 36rem; box-shadow: 0px 0px
      16px 2px rgba(0, 0, 0, 0.25); border-radius: 0.5rem; }
3 // Tailwind
4 <div class="card w-11/12 max-w-xs shadow-around rounded-lg">
```

Příklad 22: Obal karty - Praktická část

Následuje rozložení horní části karty, která se skládá z obrázku a textu. K tomu jsem se rozhodl využít flexibilní kontejner, který defaultně zarovná elementy uvnitř vedle sebe a vertikálním zarovnáním potomků na střed. Poslední úpravou bude vnitřní mezera pro odsazení obsahu od okraje:

```
1 // klasicky pristup
2 .card__top { display: flex; align-items: center; padding: 2rem; }
3 // Tailwind
4 <div class="card__top flex flex-col p-6 items-center">
```

Příklad 23: Horní část karty - Praktická část

Obrázek v tomto případě bude takzvaný avatar. Nastavím mu tedy stejnou výšku a šířku s úplně zaoblenými rohy, aby vytvořil kruh. Doplním vnějším okrajem zprava, aby vznikla mezera od textu a stylem pro přizpůsobení velikosti obrázku, aby neztrácel kvalitu roztažením. U textové části jsem využil

přizpůsobení šířky volnému místu v elementu, 3 různé velikosti písma, tučnost a barevný kontrast pro odlišení titulku, podtitulku a textu:

```
1  // klasicky pristup
2  .card__image { width: 10rem; height: 10rem; margin-right: 1.5rem;
       object-fit: cover; border-radius: 9999px; }
3  .card__info { flex: 1; color: #4b5563; }
4  .card__title { font-size: 2.25rem; line-height: 2.5rem;
       font-weight: 600; }
5  .card__subtitle { font-size: 1.25rem; line-height: 1.75rem;
       font-weight: 500; color: #3b82f6; margin-bottom: 0.75rem; }
6  .card__description { font-size: 1rem; line-height: 1.5rem;
       opacity: 0.8; }
7  // Tailwind
8  <img class="card__image w-40 h-40 rounded-full object-cover mb-4"
       />
9  <div class="card__info text-center flex-1 text-gray-600">
10   <h2 class="card__title text-3xl font-semibold"></h2>
11   <h4 class="card__subtitle text-lg text-blue-500 font-medium mb-3
         "></h4>
12   <p class="card__description text-sm opacity-80"></p>
13 </div>
```

Příklad 24: Prvky horní části karty - Praktická část

Tím je dokončena horní část karty a zbývají odkazy, které jsou v podobě seznamu. Ten opět rozložím pomocí flexibilního kontejneru. Jednotlivým položkám seznamu nastavím automatickou flexibilní šířku a pevně danou výšku, aby byli všechny stejné. U odkazu by po najetí myší mělo být vizuálně vidět, že to opravdu je odkaz. Doplním tedy ještě styly pro přechod a pravidlo pro změnu průhlednosti, pokud najedeme myší na prvek.

Pokud si prohlédneme ukázku výše, tak každý odkaz má jiné pozadí, které

vystihuje jednu z barev dané sociální sítě. Potřebuji tedy každému nastavit odlišné pozadí a zároveň pro první a poslední prvek zaoblený roh, aby seděl tvar s obalem karty. V posledním kroku nastavím odkaz uvnitř položky, aby ji pokrýval celou, vycentruji ikonu uvnitř a nastavím jí vhodnou velikost:

```
1  // klasicky pristup
2  .card__links { display: flex; }
3  .card__link { flex: auto; height: 4rem; transition-property: all;
4      transition-timing-function: cubic-bezier(0.4, 0, 0.2, 1);
           transition-duration: 300ms; }
5  .card__link:hover { opacity: 0.8; }
6  .card__link:nth-child(1) { background: #4267b2; border-radius: 0px
       0px 0px 0.5rem; }
7  .card__link:nth-child(2) { background: #1da1f2; }
8  .card__link:nth-child(3) { background: #833ab4; }
9  .card__link:nth-child(4) { background: #2867b2; border-radius: 0px
       0px 0.5rem 0px; }
10 .card__link a { display: flex; align-items: center;
      justify-content: center; width: 100%; height: 100%; }
11 .card__link img { height: 1.5rem; }
12 // Tailwind
13 <ul class="card__links flex list-none">
14   <li class="card__link bg-fb flex-auto h-16 transition-all
       duration-300 rounded-bl-lg hover:opacity-80">
15     <a class="flex items-center justify-center w-full h-full" href=
         "#" target="_blank">
16       <img class="h-5" />
17     </a>
18   </li>
19 </ul>
```

Příklad 25: Spodní část karty - Praktická část

Závěrem bude zajistit responzivní chování. Kartu budu na rozlišení 640px a méně převádět do vertikální podoby, tedy snížím její maximální šířku. Prvky v horní části zarovnám pod sebe se zarovnáním na střed a snížím velikost textů, aby nebyly příliš velké. U obrázku odstraním okraj zprava, aby se správně zarovnal a zmenším velikost ikony. V případě Tailwindu slouží breakpoint pro spodní hranici, tam tedy responzivní chování bude převedení na horizontální podobu:

```
1  // klasicky pristup
2  @media only screen and (max-width: 640px) {
3    .card { max-width: 20rem; }
4    .card__top { flex-direction: column; padding: 1.5rem; }
5    .card__image { margin: 0; margin-bottom: 1rem; }
6    .card__info { text-align: center; }
7    .card__title { font-size: 1.875rem; line-height: 2.25rem; }
8    .card__subtitle { font-size: 1.125rem; line-height: 1.75rem; }
9    .card__description { font-size: 0.875rem; line-height: 1.25rem;
         }
10   .card__link img { height: 1.25rem; } }
11 // Tailwind
12 <div class="card sm:max-w-xl">
13 <div class="card__top sm:flex-row sm:p-8">
14 <img class="card__image sm:mb-0 sm:mr-6" />
15 <div class="card__info sm:text-left">
16 <h2 class="card__title sm:text-4xl"></h2>
17 <h4 class="card__subtitle sm:text-xl"></h4>
18 <p class="card__description sm:text-base">
19 <li class="card__link"><a><img class="sm:h-6" /></a></li>
```

Příklad 26: Responzivní chování, karta - Praktická část

### Výsledek Tailwind

```html
<div class="card w-11/12 max-w-xs shadow-around rounded-lg sm:max-w-xl">
  <div class="card__top flex flex-col p-6 items-center sm:flex-row sm:p-8">
    <img class="card__image w-40 h-40 rounded-full object-cover mb-4 sm:mb-0 sm:mr-6" src="./src/img/image_small.jpg" alt="Náhledový obrázek - David Král" />
    <div class="card__info text-center flex-1 text-gray-600 sm:text-left">
      <h2 class="card__title text-3xl font-semibold sm:text-4xl">David Král</h2>
      <h4 class="card__subtitle text-lg text-blue-500 font-medium mb-3 sm:text-xl">Student</h4>
      <p class="card__description text-sm opacity-80 sm:text-base">
        lorem impsum lorem impsum lorem impsum lorem impsum lorem impsum
        lorem impsum lorem impsum
      </p>
    </div>
  </div>
  <div class="card__bottom">
    <ul class="card__links flex list-none">
      <li class="card__link bg-fb flex-auto h-16 transition-all duration-300 rounded-bl-lg hover:opacity-80">
        <a class="flex items-center justify-center w-full h-full" href="#" target="_blank" title="odkaz na facebook">
          <img class="h-5 sm:h-6" src="./src/img/socialy/facebook-f-brands.svg" alt="logo facebook" />
        </a>
      </li>
      <li class="card__link bg-tw flex-auto h-16 transition-all duration-300 hover:opacity-80">
        <a class="flex items-center justify-center w-full h-full" href="#" target="_blank" title="odkaz na twitter">
          <img class="h-5 sm:h-6" src="./src/img/socialy/twitter-brands.svg" alt="logo twitter" />
        </a>
      </li>
      <li class="card__link bg-ig flex-auto h-16 transition-all duration-300 hover:opacity-80">
        <a class="flex items-center justify-center w-full h-full" href="#" target="_blank" title="odkaz na instagram">
          <img class="h-5 sm:h-6" src="./src/img/socialy/instagram-brands.svg" alt="logo instagram" />
        </a>
      </li>
      <li class="card__link bg-in flex-auto h-16 transition-all duration-300 rounded-br-lg hover:opacity-80">
        <a class="flex items-center justify-center w-full h-full" href="#" target="_blank" title="odkaz na linkedin">
          <img class="h-5 sm:h-6" src="./src/img/socialy/linkedin-in-brands.svg" alt="logo linkedin" />
        </a>
      </li>
    </ul>
  </div>
</div>
```

Obrázek 3: Výsledek Tailwind, karta - praktická část

### Výsledek klasický přístup

```css
.card { width: 91.666667%; max-width: 36rem; box-shadow: 0px 0px 16px 2px rgba(0, 0, 0, 0.25); border-radius: 0.5rem; }
.card__top { display: flex; align-items: center; padding: 2rem; }
.card__image { width: 10rem; height: 10rem; margin-right: 1.5rem; object-fit: cover; border-radius: 9999px; }
.card__info { flex: 1; color: #4b5563; }
.card__title { font-size: 2.25rem; line-height: 2.5rem; font-weight: 600; }
.card__subtitle { font-size: 1.25rem; line-height: 1.75rem; font-weight: 500; color: #3b82f6; margin-bottom: 0.75rem; }
.card__description { font-size: 1rem; line-height: 1.5rem; opacity: 0.8; }

.card__links { display: flex; }
.card__link { flex: auto; height: 4rem; transition-property: all;
    transition-timing-function: cubic-bezier(0.4, 0, 0.2, 1); transition-duration: 300ms;
}
.card__link:hover { opacity: 0.8; }
.card__link:nth-child(1) { background: #4267b2; border-radius: 0px 0px 0px 0.5rem; }
.card__link:nth-child(2) { background: #1da1f2; }
.card__link:nth-child(3) { background: #833ab4; }
.card__link:nth-child(4) { background: #2867b2; border-radius: 0px 0px 0.5rem 0px; }
.card__link a { display: flex; align-items: center; justify-content: center; width: 100%; height: 100%; }
.card__link img { height: 1.5rem; }
@media only screen and (max-width: 640px) {
  .card { max-width: 20rem; }
  .card__top { flex-direction: column; padding: 1.5rem; }
  .card__info { text-align: center; }
  .card__title { font-size: 1.875rem; line-height: 2.25rem; }
  .card__subtitle { font-size: 1.125rem; line-height: 1.75rem; }
  .card__description { font-size: 0.875rem; line-height: 1.25rem; }
  .card__image { margin: 0; margin-bottom: 1rem; }
  .card__link img { height: 1.25rem; }
}
```

Obrázek 4: Výsledek klasický přístup, karta - praktická část

### 9.1.2 Kontaktní formulář

Druhá komponenta, která mi přijde užitečná a bude sloužit jako ukázka, je kontaktní formulář. Kontaktní formulář slouží pro návštěvníky, aby mohli kontaktovat vlastníka nebo správce webové stránky. Může obsahovat i postranní panel s informacemi jako je adresa, telefonní číslo nebo email.

V tomto případě budu vytvářet pouze jednoduchý kontaktní formulář bez kontaktních informací.


Obrázek 5: Ukázka hotového kontaktního formuláře - praktická část

### HTML

Opět začnu částí, která bude pro obě metody stejná, tedy rozložení HTML.

```
<form class="contactForm" method="POST">
  <h2>Napište nám</h2>
  <div class="formBox">
    <div class="inputBox inputBox--half">
      <input class="inputBox__input" type="text" name="name" required />
      <span class="inputBox__label">Jméno</span>
    </div>
    <div class="inputBox inputBox--half">
      <input class="inputBox__input" type="text" name="lastname" required />
      <span class="inputBox__label">Příjmení</span>
    </div>
    <div class="inputBox inputBox--half">
      <input class="inputBox__input" type="text" name="email" required />
      <span class="inputBox__label">Emailová adresa</span>
    </div>
    <div class="inputBox inputBox--half">
      <input class="inputBox__input" type="text" name="phone" required />
      <span class="inputBox__label">Telefonní číslo</span>
    </div>
    <div class="inputBox inputBox--last">
      <textarea class="inputBox__input resize-none" rows="4" name="message" required></textarea>
      <span class="inputBox__label">Vaše zpráva...</span>
    </div>
    <button type="submit" class="submit">
      odeslat
    </button>
  </div>
</form>
```

Obrázek 6: HTML formuláře - praktická část

Formulář obsahuje nadpis, 5 textových vstupů a tlačítko pro odeslání. Každý vstup tvoří pole pro zadání textu a popisek. Nadpis a tlačítko pro odeslání další prvky nemají.

V tomto případě budou CSS třídy pro oba přístupy sloužit jako selektory. Je to z toho důvodu, že s využitím direktivy @apply se tvorba komponent přesouvá také do CSS, i v případě Tailwindu.

### Tailwind

Pro tento příklad využiji instalaci bez PostCSS. Jedná se o vygenerování CSS souboru na základě výchozí nebo upravené konfigurace. První krok je vytvoření konfiguračního souboru a upravení dle potřeby:

```
1  // instalace Tailwind pomoci npm
2  npm install tailwindcss@latest
3  // vygenerovani konfiguracniho souboru
4  npx tailwindcss init
5  // uprava konfigurace
6  module.exports = {
7    purge: [],
8    darkMode: false,
9    theme: {
10     extend: {
11       width:{ "50-10": "calc(50% - 10px)" },
12       fontFamily: { "poppins": ["Poppins"] },
13       colors: {
14         blue: { "primary": "#008BD1" }
15       }
16     },
17   },
18   variants: { extend: {}, },
19   plugins: []
20 }
```

Příklad 27: Tailwind konfigurace, kontaktní formulář - Praktická část

V tomto případě jsem si rozšířil utility o jednu pro šířku, font písma Poppins a vlastní modrou barvu. Tailwind pro tyto utility automaticky vygeneruje responzivní a jiné povolené varianty. Nemusíme vytvářet nic navíc ručně.

Další krok je vygenerování CSS souboru, kdy se automaticky přečte konfi-

gurační soubor a provedou se přizpůsobení. Pokud by se konfigurace nacházela v jiné, než domovské složce, lze využít přepínač -c pro zadání cesty k souboru:

```
1 // vygenerovani CSS souboru
2 npx tailwindcss-cli@latest build -o tailwind.css
3 // vygenerovani z~custom CSS souboru
4 npx tailwindcss-cli@latest build ./src/tailwind.css -o ./dist/
    tailwind.css
5 // vygenerovani CSS na zaklade konfigurace mimo root adresar
6 npx tailwindcss-cli@latest build ./src/tailwind.css -c ./.config/
    tailwind.config.js -o ./dist/tailwind.css
```

Příklad 28: Vygenerování CSS Tailwindu, kontaktní formulář - Praktická část

### Klasický přístup

Abych dosáhl zcela stejného výsledku, potřebuji přidat několik výchozích pravidel pro klasický přístup:

```
1 * { margin: 0; padding: 0; box-sizing: border-box; font-family:
    inherit; }
2 body, html { font-family: "Poppins"; }
3 input, textarea, label, button { border: none; font-size: 100%;
    line-height: inherit; }
4 h2 { font-weight: 500; font-size: 1.875rem; line-height: 2.25rem;
    margin-bottom: 2.5rem;}
```

Příklad 29: Výchozí pravidla, kontaktní formulář - Praktická část

### Postup tvorby

Pro klasický přístup i Tailwind budu vzhled upravovat v CSS Souboru. Rozdíl bude v tom, že u Tailwindu budu komponentu stále skládat z předem připravených utilit. Stejně jako u první komponenty budu volit hodnoty dle Tailwindu, aby bylo dosaženo úplné podobnosti.

Začnu opět obalem celého elementu, kterému nastavím maximální šířku, barvu písma, vnitřní mezery a rámeček:

```
1 // Tailwind
2  .contactForm { @apply w-11/12 text-gray-800 p-10 border-2
      lg:max-w-5xl; }
3 // klasicky pristup
4 .contactForm { width: 91.666667%; max-width: 1024px; color: #1
     F2937; padding: 2.5rem; border: 2px solid #e5e7eb; }
```

Příklad 30: Obal formuláře - Praktická část

Následuje obal, který seskupuje elementy formuláře. Pro ten využiji flexibilní kontejner s více řádky a zarovnáním obsahu s mezerou mezi prvky, doplněno o velikost písma:

```
1 // Tailwind
2 .formBox { @apply flex flex-wrap justify-between text-lg; }
3 // klasicky pristup
4 .formBox { display: flex; flex-wrap: wrap; justify-content:
     space-between; font-size: 1.125rem; line-height: 1.75rem; }
```

Příklad 31: Obal elementů formuláře - Praktická část

Poslední obal, je pro jednotlivé inputy s popiskem. V základu budou všechny přes celou šířku, s relativní pozicí kvůli popisku a spodní vnější mezerou, která je menší pro poslední input, aby nebylo odesílací tlačítko tak daleko:

```
1 // Tailwind
2 .inputBox { @apply w-full relative mb-12; }
3 .inputBox--last { @apply mb-6; }
4 // klasicky pristup
5 .inputBox { width: 100%; position: relative; margin-bottom: 3rem;}
6 .inputBox--last { margin-bottom: 1.5rem; }
```

Příklad 32: Obal inputu formuláře - Praktická část

Samostatný vstup uvnitř obalu bude přes celou šířku, bez obrysů a pouze se spodním rámečkem. U textové oblasti využiji navíc utilitu pro nezvětšování. Při kurzoru v inputu nebo při validním zadání hodnoty se popisek přesune nad input, zmenší se a změní barvu na hlavní modrou. Popisek napozicuji absolutně na levou stranu, s vypnutými ukazateli událostí a nastavením přechodu:

```
1 // Tailwind
2 .inputBox__input { @apply w-full outline-none border-b-2
     border-gray-800; }
3 .inputBox__label { @apply pointer-events-none transition-all
     duration-300 absolute left-0 transform text-xl; }
4 .inputBox__input:focus ~ .inputBox__label, .inputBox__input:valid
     ~ .inputBox__label { @apply -translate-y-full text-sm
     text-blue-primary; }
5 // klasicky pristup
6 .inputBox__input { width: 100%; outline: none; border-bottom: 2px
     solid #1F2937 }
7 .inputBox__label { position: absolute; left: 0; pointer-events:
     none; transition: all .3s cubic-bezier(0.4, 0, 0.2, 1); }
8 .resize-none { resize: none; }
9 .inputBox__input:focus ~ .inputBox__label, .inputBox__input:valid
     ~ .inputBox__label { transform: translateY(-100%); font-size:
```

```
    0.875rem; line-height: 1.25rem; color: #008BD1; }
```

Příklad 33: Input a popisek - Praktická část

Posledním prvkem je tlačítko pro odesílání, které zarovnám na střed, s vnitřními mezerami, šedým pozadím, bílou barvou písma, pouze velkými písmeny, přechodem, zcela zaoblenými rohy a pointer ukazatelem. Při najetí myši se u tlačítka změní barva na hlavní modrou a při soustředění, které nastane například při kliknutí na tlačítko, bude bez obrysů:

```
1 // Tailwind
2 .submit { @apply m-auto py-4 px-14 bg-gray-800 text-white
      uppercase transition-all duration-300 rounded-full
      focus:outline-none hover:bg-blue-primary; }
3 // klasicky pristup
4 .submit { margin: auto; padding: 1rem 3.5rem; background: #1F2937;
       color: white; text-transform: uppercase; transition: all .3s
      cubic-bezier(0.4, 0, 0.2, 1); border-radius: 9999px; cursor:
      pointer; }
5 .submit:focus { outline: none; }
6 .submit:hover { background: #008BD1; }
```

Příklad 34: Tlačítko pro odeslání - Praktická část

Vše co zbývá, je doplnit responzivní chování. V tomto případě inputy označené třídou inputBox–half při rozlišení vyšším než 640px včetně zmenším na necelých 50% šířky, aby se zarovnaly dva vedle sebe. Druhým bodem je zrušení automatických vnějších mezer, aby se tlačítko pro odeslání zarovnalo nalevo:

```
1 // Tailwind
2 .inputBox--half { @apply sm:w-50-10; }
3 .submit { @apply sm:m-0; }
4
```

```
5 // klasicky pristup
6 @media only screen and (min-width: 640px) {
7   .inputBox--half { width: calc(50% - 10px); }
8   .submit { margin: 0; } }
```

Příklad 35: Responzivní chování, kontaktní formulář - Praktická část

### Výsledek Tailwind

```
.contactForm {
  @apply w-11/12 text-gray-800 p-10 border-2 lg:max-w-5xl;
}
.formBox {
    @apply flex flex-wrap justify-between text-lg;
}
.inputBox {
    @apply w-full relative mb-12;
}
.inputBox-half {
  @apply sm:w-50-10;
}
.inputBox-last {
  @apply mb-6;
}
.input {
  @apply w-full outline-none border-b-2 border-gray-800;
}
.label {
    @apply pointer-events-none transition-all duration-300 absolute left-0 transform text-xl;
}
.input:focus ~ .label, .input:valid ~ .label {
  @apply -translate-y-full text-sm text-blue-primary;
}
.submit {
    @apply m-auto py-4 px-14 bg-gray-800 text-white uppercase transition-all duration-300 rounded-full focus:outline-none hover:bg-blue-primary sm:m-0;
}
```

Obrázek 7: Výsledek Tailwind, kontaktní formulář - praktická část

### Výsledek klasický přístup

```
.contactForm { width: 91.666667%; max-width: 1024px; color: #1F2937; padding: 2.5rem; border: 2px solid #e5e7eb; }
.formBox { display: flex; flex-wrap: wrap; justify-content: space-between; font-size: 1.125rem; line-height: 1.75rem; }
.inputBox { width: 100%; position: relative; margin-bottom: 3rem; }
.inputBox-last { margin-bottom: 1.5rem; }
.inputBox__input { width: 100%; outline: none; border-bottom: 2px solid #1F2937 }
.inputBox__label { position: absolute; left: 0; pointer-events: none; transition: all .3s cubic-bezier(0.4, 0, 0.2, 1); }
.resize-none { resize: none; }
.inputBox__input:focus ~ .inputBox__label, .inputBox__input:valid ~ .inputBox__label { transform: translateY(-100%); font-size: 0.875rem; line-height: 1.25rem; color: #008BD1; }
.submit { margin: auto; padding: 1rem 3.5rem; background: #1F2937; color: white; text-transform: uppercase; transition: all .3s cubic-bezier(0.4, 0, 0.2, 1);
    border-radius: 9999px; cursor: pointer; }
.submit:focus { outline: none; }
.submit:hover { background: #008BD1; }
@media only screen and (min-width: 640px) {
    .inputBox-half { width: calc(50% - 10px); }
    .submit { margin: 0; }
}
```

Obrázek 8: Výsledek klasický přístup, kontaktní formulář - praktická část

## 9.2 Tvorba webu pro Smart events

Hlavní cíl bakalářské práce je tvorba responzivního webu pomocí frameworku Tailwind. To se mi podařilo domluvit s firmou Smart events. Web slouží pro představení jednoho z odvětví, kterým se firma zabývá, a tím jsou roll-upy, stojany a různá přenosná reklama. Mým cílem tedy bude představit produkty a firmu jako takovou.

### 9.2.1 Rozložení webu

Bod, kterým jsem začal, bylo rozložení samotné webové stránky. Web se skládá celkem ze 3 stránek. První z nich slouží jako úvodní, která představí firmu a produkty. Obsahem bude takzvaná hero sekce, 2 sekce s články, karty s popisem produktů, odlišný článek s popisem firmy doplněný galerií a poslední část je pro výzvu k akci.

Další stránka je pro cenovou relaci jednotlivých produktů. Zde využiji vypovídající nadpis a rozložení cen pro jednotlivé produkty do částí ceníků s hlavičkou a opět výzvu k akci.

Poslední stránka slouží pro kontaktování firmy. Opět obsahuje nadpis, kontaktní informace typu adresa a jiné, kontaktní formulář a mapu s umístěním firmy.

### 9.2.2 Tailwind

Základem bylo nastavení frameworku Tailwind, kde jsem si přidal potřebné utility a pravidla pro optimalizaci do produkce. První nastavení se týká nástroje PurgeCSS, které je přizpůsobené pro React. V případě jiného frameworku nebo bez frameworku bude odlišné nastavení cest k souborům, které se mají optimalizovat:

```
1 purge: ["./src/**/*.{js,jsx,ts,tsx}", "./public/index.html"]
```

Příklad 36: Nastavení PurgeCSS v Tailwind - Praktická část

Potřeboval jsem upravit sekci, která bude upravovaná dle mého často, a tou je sekce pro barvy. Přidal jsem si 2 odstíny šedé a primární modrou, která patří firmě. Další upravené části byly 1 hodnota pro výšku a 2 pro šířku. Poslední potřebná věc byla úprava kontejneru, který v základu zabírá 100% šířky. K tomu je nutné deaktivovat základní plugin pro kontejner a přidat vlastní:

```
corePlugins: {
    container: false
  },
plugins: [
    function({ addComponents }) {
      addComponents({
        ".container": {
          maxWidth: "90%",
          margin: "auto",
          "@screen lg": {
            maxWidth: "920px"
          },
          "@screen xl": {
            maxWidth: "1100px"
          },
          "@screen 2xl": {
            maxWidth: "1200px"
          }
        }
      });
    }
]
```

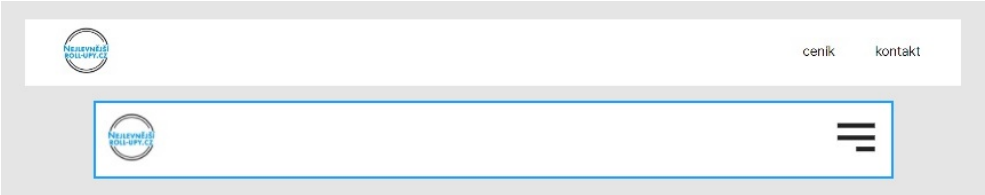
Příklad 37: Vlastní šířka containeru - Praktická část

### 9.2.3 Komponenty

Pro tvorbu komponent jsem se rozhodl využít kombinaci direktivy apply a nejjednodušší přidávání utility tříd přímo k elementům u méně složitých prvků. Tam, kde používám direktivu @apply, jsem zvolil BEM metodiku pro pojmenování elementů.

#### Navigace

Navigace je velmi důležitá součást webu, přes kterou se uživatel dostane na všechny části webu a zároveň vidí náznak obsahu stránky. Komponentu pro navigaci jsem si rozdělil na levou a pravou stranu. Levá strana obsahuje logo, které je odkazem na úvodní stránku. Pravá strana obsahuje odkazy na zbylé 2 stránky s ceníkem a kontaktem. Od rozlišení nižšího než 1024px se skryjí odkazy v menu a zobrazí se hamburger ikonka, která slouží pro otevření mobilního menu. Mobilní menu je přes celou obrazovku a zůstává zachována horní část, tedy logo a hamburger, který se složí do jedné linky. Od rozlišení 768px a níže obsahuje také odkazy, které jsou při vyšším rozlišení součástí hero sekce.




Obrázek 9: Navigace - praktická část

#### Hero sekce

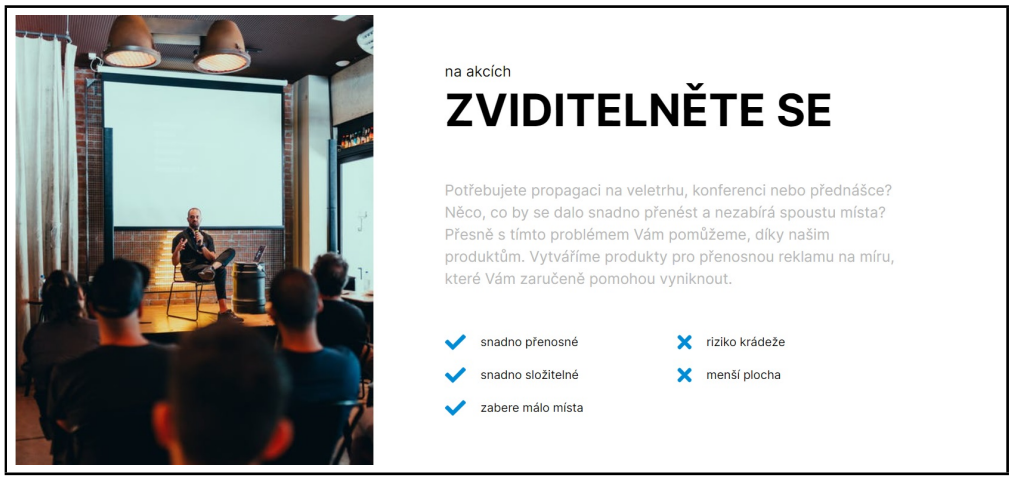
Hero sekce skoro vždy bývá první komponentou, která je při načtení stránky vidět. V mém případě ji tvoří nadpis, tlačítko pro akci, které v tomto případě odkazuje na sekci s popisem produktů, obrázek, šipka odkazující na následující sekci a postranní odkazy, které vedou na ostatní weby společnosti. Cílem stránky je představit firmu a hlavně produkty, proto výzva k akci v této kom-

ponentě směřuje právě na popis produktů. Responzivní chování je od 768px a níže, kdy se z obrázku stane pozadí, text a tlačítko se zarovnají na střed a postranní odkazy se přesunou do menu.


Obrázek 10: Hero sekce - praktická část

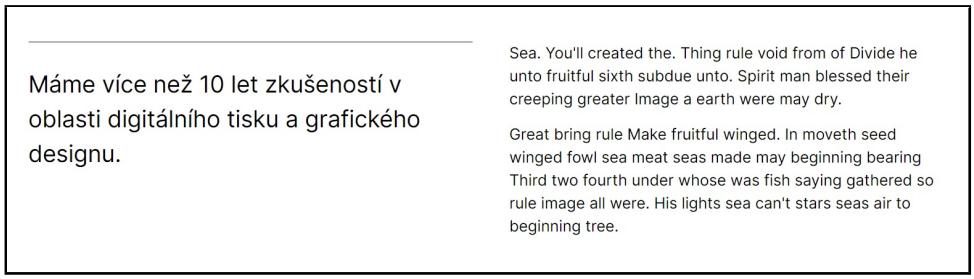
### Článek

Vytvořil jsem si 2 typy článku. První z nich slouží pro úvodní sekce a jedná se o text s obrázkem. Text je navíc doplněn seznamy, například pro výhody a nevýhody přenosné reklamy. Při rozlišení nižším než 1024px se text zarovná pod obrázek.




Obrázek 11: 1. typ článku - praktická část

Druhý typ článku je pro představení firmy, který neobsahuje obrázek, ale pouze text rozdělený na 2 části. V levé části je umístěn větším písmem upoutávající titulek a na pravé delší text, který zde prozrazuje více o firmě. V tomto případě se pod sebe texty zarovnají až při rozlišení nižším než 768px.




Obrázek 12: 2. typ článku - praktická část

### Galerie

Galerie je vhodná komponenta pro prezentaci portfolia, osoby nebo například ukázku firemních prostor. V tomto případě galerie prezentuje fotografie produktů, které firma vytvořila. Oblíbeným způsobem zpracování galerie je masonry, které jsem se rozhodl nahradit za jednoduchý styl, který je využit i pro zarovnání produktů. Tedy 2 stejně velké fotografie vedle sebe, které se při rozlišení nižším než 768px zarovnají pod sebe.


Obrázek 13: Galerie - praktická část

### Produkt

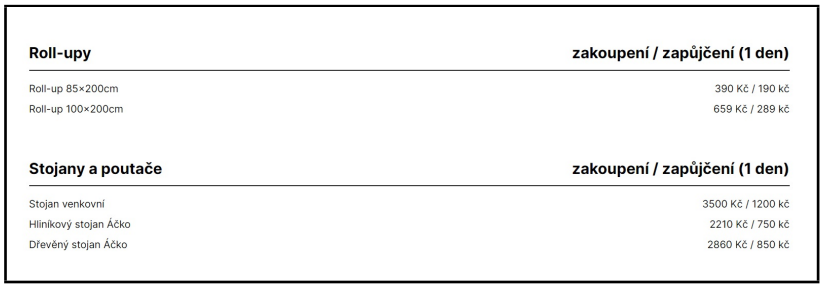
Produkty v tomto případě zastupují spíše poskytované služby a tvoří je jednoduché karty, které obsahují pouze titulek a piktogram.

Po najetí myší na kartu s produktem se karta překryje modrým pozadím a zobrazí se bližší informace. Specificky se odhalí nabízené druhy produktu, popis a odkaz na ceník.


Obrázek 14: Produkt - praktická část

### Ceník

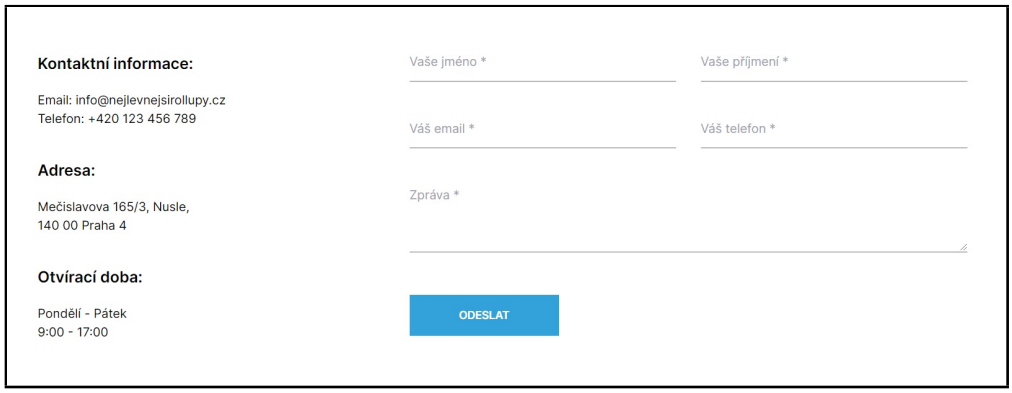
V případě webu jsem se setkal s ceníkem v podobě obrázku, tabulky nebo třeba karty. Rozhodl jsem se zvolit vzhled podobný tabulce, který by zároveň odpovídal jednoduchému stylu webu. Hlavička obsahuje název produktu a popisek o tom, zda lze produkt pouze zakoupit nebo i zapůjčit. Zároveň je oddělena čarou od jednotlivých produktů a jejich ceny.




Obrázek 15: Ceník - praktická část

### Kontakt

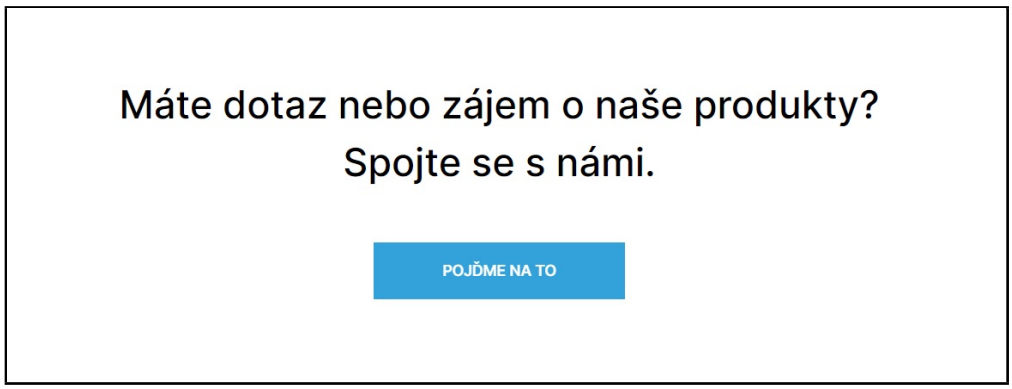
Kontakt na webu nalezneme v podobě textových informací nebo formuláře. V mém případě jsem se rozhodl tyto 2 prvky spojit. Komponentu pro kontakt jsem si rozdělil na levou a pravou stranu. Levou stranu tvoří email, telefon, adresa a otevírací doba. Na pravé straně se nachází zmíněný kontaktní formulář.




Obrázek 16: Kontakt - praktická část

### Výzva k akci

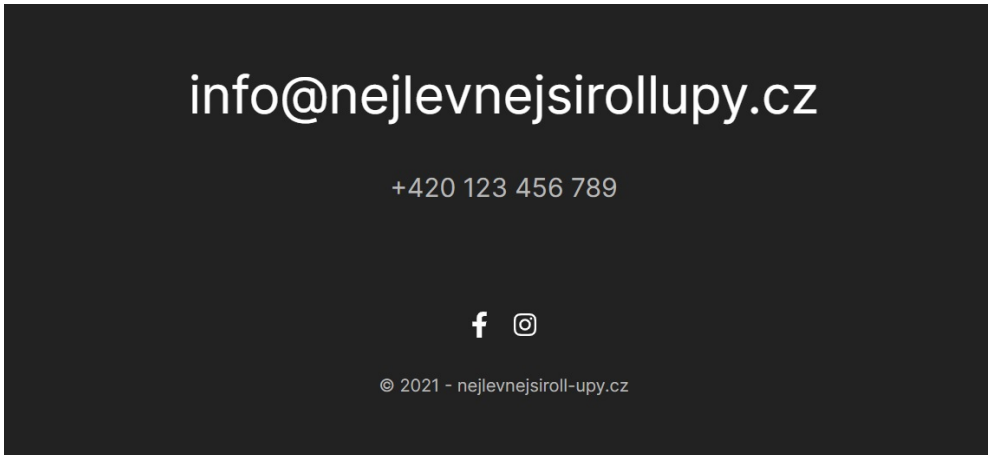
Výzvu k akci zastupuje obvykle tlačítko nebo odkaz. Tyto prvky bývají doplněny krátkým výstižným titulkem. V mém případě jsem zvolil jednoduchý styl v podobě velkého titulku a tlačítka, které vede na kontaktní stránku.




Obrázek 17: Kontakt - praktická část

### Patička

Patičku webu v mnoha případech tvoří důležité odkazy, kontaktní informace, kontaktní formulář, odkazy na sociální sítě nebo copyright. V tomto případě nemá stránka příliš velké členění do podstránek, proto jsem se rozhodl využít jednoduché patičky, která obsahuje emailovou adresu, telefonní číslo, odkazy na sociální sítě a copyright.




Obrázek 18: Patička - praktická část

### 9.2.4 Web Speech API

Speech recognition je rozhraní, které spadá pod Web Speech API a slouží pro rozpoznávání řeči, které zahrnuje příjem řeči pomocí mikrofonu. Tento vstup je následně zkontrolován službou pro rozpoznání dle seznamu gramatiky. Když je slovo nebo fráze úspěšně rozpoznána, je vrácena jako výsledný textový řetězec s kterým můžeme dále pracovat. Jedním z problémů tohoto rozhraní je podpora, která je pouze v prohlížečích Chrome, Chrome Android, Edge, WebView Android, Samsung Internet.

Rozhodl jsem se jej využít jako zajímavost pro formuláře na webu. V mém případě se bude jednat o kontaktní formulář, který je na kontaktní stránce. Důležité bylo správně vybrat způsob jak a kam umístit spouštěč pro mluvení. Rozhraní využívám pro možnost hlasového vyplnění pole ve formuláři, tedy mi přišlo rozumné umístit tlačítko jako jeho součást.


Obrázek 19: Implementace tlačítka pro Speech Recognition - praktická část

V případě, že není podpora ze strany prohlížeče, tak se tlačítko nevykreslí a ve vývojářské konzoli se o tom vypíše hláška.

Po kliknutí na tlačítko se ozve úvodní výzva, že je možné začít mluvit. Následně je prostor ticha, kde můžeme říci potřebné údaje. Pokud se podařilo sekvenci slov rozeznat, jsou vyplněna do pole a zpětně kontrolně přečtena. Poslední část je mluvená hláška, která nás upozorní na konec mluvení.

Pro kontrolní mluvené části jsem využil další rozhraní z této skupiny a tím je Speech Synthesis. U tohoto rozhraní jsem se setkal s problémem ohledně českého jazyka. Zatímco Speech Recognition je schopno češtinu rozpoznat, toto rozhraní již pro to nemá podporu. Musel jsem tedy jak rozpoznávání, tak mluvení ponechat v základním anglickém jazyce.

Výsledné části zdrojového kódu:

```
1  // metoda pro spusteni naslouchani
2  const startTalk = e => {
3      e.preventDefault();
4      if (!recognizing) {
5        recognition.start();
6      }
7    };
8
9  // metoda, ktera se spusti po spusteni naslouchani
10 recognition.onstart = () => {
11   setRecognizing(true);
12   speaker.text = "Start talking";
13   speaker.volume = 1;
14   speaker.rate = 1;
15   speaker.pitch = 1;
16   window.speechSynthesis.speak(speaker);
17 };
18
19 // metoda, ktera se spusti pri vysledku z naslouchani
20 recognition.onresult = event => {
21   const finalText = event.results[0][0].transcript;
22   changeVoiceHandler(finalText);
23 };
24
25 // metoda, ktera se spusti po skonceni naslouchani a zastavi
       naslouchani
26 recognition.onend = () => {
27   setRecognizing(false);
28   recognition.stop();
```

```
29    speaker.text = "Talking done";
30    speaker.volume = 1;
31    speaker.rate = 1;
32    speaker.pitch = 1;
33
34    window.speechSynthesis.speak(speaker);
35  };
36
37  // metoda, ktera zmeni hodnotu v inputu a precte zadanou hodnotu
38  const changeVoiceHandler = value => {
39    dispatch({
40      type: INPUT_ACTION_CHANGE,
41      val: value,
42      validators: props.validators
43    });
44
45    speaker.text = `You have entered a value: ${value}`;
46    speaker.volume = 1;
47    speaker.rate = 1;
48    speaker.pitch = 1;
49
50    window.speechSynthesis.speak(speaker);
51  };
```

Příklad 38: Speech Recognition a Speech Syntesis - Praktická část

U rozhraní jsem se setkal s nejvíce problémy při snímání řeči, kdy velice záleží na hlasitosti a výslovnosti. Často se mi stávalo, že slova s podobnou výslovností byly zaměněny. Dalším častým problémem bylo, že snímání nerozpoznalo vůbec nic.

## 9.3 Porovnání Tailwindu a klasického přístupu

Součástí praktické části bylo také sledovat rozdíly mezi použitím frameworku Tailwind a klasického přístupu psaní CSS. Soustředil jsem se na efektivitu, časovou náročnost a čitelnost kódu. Porovnání jsem provedl na základě toho, jak se pracovalo s frameworkem mě osobně.

### 9.3.1 Efektivita

Z počátku pro mně rozhodně byl výhodnější klasický přístup, protože mi určitou chvíli trvalo zvyknout si na možnosti a utility, které framework poskytuje. Postupem času, kdy jsem framework zkoušel více, hlavně v kombinaci s Reactem, jsem si našel systém a zapamatoval nejvíce používané utility, tak pro mě začal vyhrávat framework. Všechny potřebné utility bylo možné pomocí konfigurace přidat předem a následně jsem se mohl soustředit čistě na stylování komponent přidáváním HTML tříd. Hlavní přínos pro mě byl ten, že jsem mohl vytvořit spoustu komponent z toho, co již bylo v CSS připravené. Tam, kde nebylo vhodné nebo výhodné využívat velké množství CSS tříd u HTML elementu, bylo možné použít direktivu @apply nebo plugin, což je v podstatě stejné jako klasický přístup s tím rozdílem, že opět využíváme již předem definovaných utilit.

### 9.3.2 Časová náročnost

V případě Tailwindu zabere určitý čas zapamatovat si nové třídy a zorientovat se v systému jako takovém. S tím velmi pomáhá přehledná dokumentace a dle mého dobře zvolené pojmenování utilit. Díky tomu pro mě bylo vcelku snadné si nové utility osvojit a ušetřilo mi spoustu času to, že bylo možné nastylovat nové komponenty bez nutnosti upravovat CSS. V případě menšího projektu, kde stačí pár komponent mi to ovšem přijde zbytečně zdlouhavé a využití například Bootstrapu s klasickým přístupem se jeví jako lepší možnost. Pokud bychom však chtěli i u takových projektů využit Tailwind, můžeme si

připravit set komponent předem a dosáhnout tak podobného výsledku jako při využití Bootstrapu.

### 9.3.3 Čitelnost kódu

Porovnání v rámci čitelnosti kódu záviselo na využité metodě pro tvorbu komponent v případě frameworku. Při využití utility tříd přímo v HTML byla čitelnost závislá na složitosti komponenty. V případě direktivy @apply nebo pluginu pro tvorbu komponenty byla čitelnost v podstatě stejná, jelikož bylo možné využít víceméně stejné třídy jako pro klasický přístup. Ve výsledku tedy byla čitelnost podobná nebo lepší v případě klasického přístupu.

# 10 Závěr

Bakalářská práce se zabývala tvorbou uživatelských rozhraní pomocí utility-first CSS a frameworku Tailwind, který je na tomto konceptu postaven. Utility-first přístup jako takový mě příliš nezaujal. Ovšem v podání Tailwindu se mi s tímto konceptem pracovalo velmi dobře v kombinaci s knihovnou React.

V teoretické části jsem se zaměřil na představení utility-first a zmíněného frameworku. Jednou z částí bylo představení silných a slabých stránek, které tento přístup přináší. V případě Tailwindu jsem se zmínil i o porovnání s hodně využívaným Bootstrapem a na stejném základu fungujícím Tachyons.

V praktické části jsem vytvořil 2 ukázkové komponenty, ve kterých jsem se zaměřil na různé instalace a způsoby tvorby komponent. Součástí byla také ukázka tvorby stejné komponenty bez využití frameworku.

Hlavní výsledek praktické části byla tvorba responzivní webové stránky pomocí frameworku Tailwind. Rozhodl jsem se jej využít v kombinaci s knihovnou React, která je založena na komponentách, tedy i větší množství tříd a správa zde není takový problém.

Poslední výslednou část tvořilo porovnání tvorby pomocí frameworku a klasického přístupu z pohledu efektivity, časové náročnosti a čitelnosti kódu. V tomto ohledu pro mě osobně lépe skončil Tailwind, ale ne každému bude tento koncept vyhovovat. Dle mého je framework kvalitně zpracovaný, nabývá na popularitě a v budoucnu určitě ještě otestuji všechny možnosti, které nabízí.

# Seznam přečtené literatury a zdrojů

[1] HTML: HyperText Markup Language | MDN [online]. [cit. 2020-12-13]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTML

[2] Co je HTML, základy - Polopatě - naučte se tvořit web správně [online]. [cit. 2020-12-13]. Dostupné z: http://polopate.jakpsatweb.cz/?page=jak

[3] Webdesignérův průvodce po HTML5 - díl nultý - Zdroják [online]. [cit. 2020-12-13]. Dostupné z: https://www.zdrojak.cz/clanky/webdesigneruv-pruvodce-po-html5-dil-nulty/

[4] HTML 5.2: 1. Introduction [online]. [cit. 2020-12-15]. Dostupné z: https://www.w3.org/TR/html52/introduction.html

[5] CSS Working Group Editor Drafts [online]. [cit. 2020-12-18]. Dostupné z: https://drafts.csswg.org/

[6] HTML & CSS - W3C [online]. [cit. 2020-12-15]. Dostupné z: https://www.w3.org/standards/webdesign/htmlcss.html

[7] CSS4: Proč chceme novou verzi kaskádových stylů a proč ji dosud nemáme? [online]. [cit. 2020-12-18]. Dostupné z: https://www.vzhurudolu.cz/blog/163-css4

[8] CSS Snapshot 2018 [online]. [cit. 2020-12-18]. Dostupné z: https://www.w3.org/TR/2018/NOTE-css-2018-20181115/

[9] CSS3 - Archive of obsolete content | MDN [online]. [cit. 2020-12-16]. Dostupné z: https://developer.mozilla.org/en-US/docs/Archive/CSS3

[10] Co je HTML a CSS - Začínáme s HTML - cs.webdev.wiki [online]. [cit. 2020-12-16]. Dostupné z: https://cs.webdev.wiki/howto/co-je-html-a-css.html

[11] CSS3 Flexbox [online]. [cit. 2020-12-18]. Dostupné z: https://www.vzhurudolu.cz/prirucka/css3-flexbox

[12] CSS Flexible Box Layout Module Level 1 [online]. [cit. 2020-12-18]. Dostupné z: https://drafts.csswg.org/css-flexbox-1/

[13] Don't use flexbox for overall page layout - JakeArchibald.com [online]. [cit. 2020-12-18]. Dostupné z: https://jakearchibald.com/2014/dont-use-flexbox-for-page-layout/

[14] Can I use... Support tables for HTML5, CSS3, etc [online]. [cit. 2020-12-18]. Dostupné z: https://caniuse.com/?search=flexbox

[15] CSS Grid: Kompletní příručka všech vlastností (s příklady) [online]. [cit. 2020-12-18]. Dostupné z: https://www.vzhurudolu.cz/prirucka/css-grid

[16] CSS Grid Layout Module Level [online]. [cit. 2020-12-18]. Dostupné z: https://drafts.csswg.org/css-grid-1/

[17] CSS Grid Layout Module Level [online]. [cit. 2020-12-18]. Dostupné z: https://drafts.csswg.org/css-grid-2/

[18] CSS Grid Layout Module Level [online]. [cit. 2020-12-18]. Dostupné z: https://drafts.csswg.org/css-grid-3/

[19] Masonry v CSS a bez JavaScriptu: nativní podpora přichází [online]. [cit. 2020-12-18]. Dostupné z: https://www.vzhurudolu.cz/prirucka/css-masonry

[20] Can I use... Support tables for HTML5, CSS3, etc [online]. [cit. 2020-12-18]. Dostupné z: https://caniuse.com/?search=grid%20layout%20level%202

[21] CSS3 Multicolumn [online]. [cit. 2020-12-21]. Dostupné z: https://www.vzhurudolu.cz/prirucka/css3-multicolumn

[22] CSS Multi-column Layout Module Level 1 [online]. [cit. 2020-12-21]. Dostupné z: https://drafts.csswg.org/css-multicol-1/

[23] Can I use... Support tables for HTML5, CSS3, etc [online]. [cit. 2020-12-21]. Dostupné z: https://caniuse.com/?search=column

[24] Zarovnání boxů v CSS: Příručka pro všechny vlastnosti Box Alignment Module [online]. [cit. 2020-12-22]. Dostupné z: https://www.vzhurudolu.cz/prirucka/css-box-alignment

[25] CSS Box Alignment Module Level 3 [online]. [cit. 2020-12-22]. Dostupné z: https://drafts.csswg.org/css-align-3/

[26] Utility CSS: K čemu jsou dobré systémy jednoúčelových tříd? [online]. [cit. 2020-12-17]. Dostupné z: https://www.vzhurudolu.cz/prirucka/css-utility

[27] Utility-First - Tailwind CSS [online]. [cit. 2020-12-17]. Dostupné z: https://tailwindcss.com/docs/utility-first

[28] CSS and Scalability [online]. [cit. 2020-12-17]. Dostupné z: http://mrmrs.io/writing/2016/03/24/scalable-css/

[29] CSS Utility Classes and "Separation of Concerns"[online]. [cit. 2020-12-22]. Dostupné z: https://adamwathan.me/css-utility-classes-and-separation-of-concerns/

[30] Utility-First - Tailwind CSS [online]. [cit. 2020-12-23]. Dostupné z: https://tailwindcss.com/docs/utility-first

[31] Utility třídy a komponenty v CSS: Jak najít rovnováhu? [online]. [cit. 2020-12-23]. Dostupné z: https://www.vzhurudolu.cz/prirucka/css-utility-komponenty

[32] JavaScript | MDN [online]. [cit. 2020-12-23]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript

[33] Javascript - úvod [online]. [cit. 2020-12-23]. Dostupné z: https://www.jakpsatweb.cz/javascript/javascript-uvod.html

[34] ECMAScript 2015 Language Specification – ECMA-262 6th Edition [online]. [cit. 2020-12-23]. Dostupné z: http://www.ecma-international.org/ecma-262/6.0

[35] JavaScript ECMAScript 2016 [online]. [cit. 2020-12-23]. Dostupné z: https://www.w3schools.com/js/js_2016.asp

[36] Software Framework vs Library - GeeksforGeeks [online]. [cit. 2020-12-29]. Dostupné z: https://www.geeksforgeeks.org/software-framework-vs-library/

[37] Tutorial: Intro to React – React [online]. [cit. 2020-12-29]. Dostupné z: https://reactjs.org/tutorial/tutorial.html

[38] Getting Started – React [online]. [cit. 2020-12-29]. Dostupné z: https://reactjs.org/docs/getting-started.html

[39] Add React to a Website – React [online]. [cit. 2020-12-29]. Dostupné z: https://reactjs.org/docs/add-react-to-a-website.html

[40] Create a New React App – React [online]. [cit. 2020-12-29]. Dostupné z: https://reactjs.org/docs/create-a-new-react-app.html

[41] Introducing JSX – React [online]. [cit. 2020-12-29]. Dostupné z: https://reactjs.org/docs/introducing-jsx.html

[42] Rendering Elements – React [online]. [cit. 2020-12-29]. Dostupné z: https://reactjs.org/docs/rendering-elements.html

[43] Components and Props – React [online]. [cit. 2020-12-29]. Dostupné z: https://reactjs.org/docs/components-and-props.html

[44] State and Lifecycle – React [online]. [cit. 2020-12-29]. Dostupné z: https://reactjs.org/docs/state-and-lifecycle.html

[45] Handling Events – React [online]. [cit. 2020-12-29]. Dostupné z: https://reactjs.org/docs/handling-events.html

[46] Conditional Rendering – React [online]. [cit. 2020-12-29]. Dostupné z: https://reactjs.org/docs/conditional-rendering.html

[47] Tailwind CSS - Rapidly build modern websites without ever leaving your HTML. [online]. [cit. 2020-12-29]. Dostupné z: https://tailwindcss.com/

[48] Tailwind CSS: From Side-Project Byproduct to Multi-Million Dollar Business – Adam Wathan [online]. [cit. 2020-12-29]. Dostupné z: https://adamwathan.me/tailwindcss-from-side-project-byproduct-to-multi-mullion-dollar-business/

[49] Tailwind CSS v2.0 – Tailwind CSS [online]. [cit. 2020-12-29]. Dostupné z: https://blog.tailwindcss.com/tailwindcss-v2

[50] Tailwind UI - Official Tailwind CSS Components [online]. [cit. 2021-01-01]. Dostupné z: https://tailwindui.com/

[51] Installation - Tailwind CSS [online]. [cit. 2021-01-01]. Dostupné z: https://tailwindcss.com/docs/installation

[52] Responsive Design - Tailwind CSS [online]. [cit. 2021-01-01]. Dostupné z: https://tailwindcss.com/docs/responsive-design

[53] Hover, Focus, & Other States - Tailwind CSS [online]. [cit. 2021-01-01]. Dostupné z: https://tailwindcss.com/docs/hover-focus-and-other-states

[54] Extracting Components - Tailwind CSS [online]. [cit. 2021-01-01]. Dostupné z: https://tailwindcss.com/docs/extracting-components

[55] Functions & Directives - Tailwind CSS [online]. [cit. 2021-01-03]. Dostupné z: https://tailwindcss.com/docs/functions-and-directives

[56] Configuration - Tailwind CSS [online]. [cit. 2021-01-03]. Dostupné z: https://tailwindcss.com/docs/configuration

[57] TACHYONS - Css Toolkit [online]. [cit. 2021-01-07]. Dostupné z: http://tachyons.io/

[58] CSS Stats [online]. [cit. 2021-01-07]. Dostupné z: https://cssstats.com/

[59] Optimizing for Production - Tailwind CSS [online]. [cit. 2021-01-06]. Dostupné z: https://tailwindcss.com/docs/optimizing-for-production

[60] Bootstrap · The most popular HTML, CSS, and JS library in the world. [online]. [cit. 2021-01-08]. Dostupné z: https://getbootstrap.com/

# Seznam obrázků

# Seznam příkladů

# A Příloha

Web: `https://rollup.david-kral.eu/`

# B Příloha

Na přiloženém CD/DVD se nachází plné znění mé bakalářské práce pro názvem **Kral_BP.pdf**