Pedagogická fakulta
Faculty of Education

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

Jihočeská univerzita v ČB

Pedagogická fakulta

Katedra informatiky

Rozpoznávání hlasu a převod na text pomocí Speech Recognition JS API

Voice recognition and text conversion using the Speech Recognition JS API

Bakalářská práce

Vypracoval: Martin Novák

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

České Budějovice 2023

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH

Pedagogická fakulta

Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Martin NOVÁK**
Osobní číslo:	**P20495**
Studijní program:	**B7507 Specializace v pedagogice**
Studijní obor:	**Informační technologie a e-learning**
Téma práce:	**Rozpoznávání hlasu a převod na text pomocí Speech Recognition JS API**
Zadávající katedra:	**Katedra informatiky**

Zásady pro vypracování

Cílem bakalářské práce je zpracovat problematiku rozpoznávání hlasu a hlasového zadávání textu pomocí JavaScriptového rozhraní SpeechRecognition. Toto rozhraní je součástí Web Speech API, které umožňuje začlenit hlasová data do webových aplikací a obsahuje dvě komponenty – převod textu na řeč (SpeechSynthesis) a asynchronní rozpoznávání řeči (SpeechRecognition). Technologie JS API umožňuje rozpoznávat hlas přicházející na vstup, následně ho převést na text, který lze pak lze využít mnoha způsoby, například pouhým zápisem jako diktafon, či vyvoláváním dalších funkcí pomocí slovního příkazu. Velmi pozitivní a především aktuální vlastností je i skutečnost, že se jedná o tzv. blind friendly techniku, tedy technologii určenou pro v tomto případě zrakově handicapované uživatele webových aplikací.
V teoretické části se autor zaměří na představení základních webových technologií souvisejících s problematikou se zaměřením na JavaScript a především na samotné SpeechRecognition API a jeho implementaci a zakomponování pomocí již zmíněného JavaScriptu. V praktické části bude vytvořena webová aplikace, která bude demonstrovat možnosti, vlastnosti a výhody či nevýhody, které tato technologie rozpoznávání hlasu v současné době přináší.

Rozsah pracovní zprávy:	**40**
Rozsah grafických prací:	**interaktivní modely**
Forma zpracování bakalářské práce:	**tištěná**

Seznam doporučené literatury:

1. MDN Web Docs. Speech Recognition [online]. 1 [cit. 2022-03-30]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/SpeechRecognition
2. Kai Wedekind. Speech Recognition API [online]. 1 [cit. 2022-03-30]. Dostupné z: https://codeburst.io/html5-speech-recognition-api-670846a50e92
3. Mohan Raj. Speech Recognition Using the Web Speech API in JavaScript [online]. 1 [cit. 2022-03-30]. Dostupné z: https://www.section.io/engineering-education/speech-recognition-in-javascript/
4. Zdroják. Rozpoznávání hlasu [online]. 1 [cit. 2022-03-30]. Dostupné z: https://zdrojak.cz/clanky/rozpoznavani-hlasu/
5. IBM. What is speech recognition? [online]. 1 [cit. 2022-03-30]. Dostupné z: https://www.ibm.com/cloud/learn/speech-recognition
6. TechTarget. What is speach recognition? [online]. 1 [cit. 2022-03-30]. Dostupné z: https://www.techtarget.com/searchcustomerexperience/definition/speech-recognition

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

Datum zadání bakalářské práce: **4. dubna 2022**
Termín odevzdání bakalářské práce: **30. dubna 2023**

L.S.

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

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

V Českých Budějovicích dne 4. dubna 2022

Prohlášení

Prohlašuji, že jsem autorem této kvalifikační práce a že jsem ji vypracoval pouze s použitím pramenů a literatury uvedených v seznamu použitých zdrojů.

V Českých Budějovicích dne 16. dubna 2023.

Martin Novák

Abstrakt/Anotace

Cílem bakalářské práce je zpracovat problematiku rozpoznávání hlasu a hlasového zadávání textu pomocí JavaScriptového rozhraní SpeechRecognition. Toto rozhraní je součástí Web Speech API, které umožňuje začlenit hlasová data do webových aplikací a obsahuje dvě komponenty - převod textu na řeč (SpeechSynthesis) a asynchronní rozpoznávání řeči (SpeechRecognition). Technologie JS API umožňuje rozpoznávat hlas přicházející na vstup, následně ho převést na text, který pak lze využít mnoha způsoby, například pouhým zápisem jako diktafon, či vyvoláváním dalších funkcí pomocí slovního příkazu. Velmi pozitivní, a především aktuální vlastností je i skutečnost, že se jedná o tzv. blind friendly techniku, tedy technologii určenou pro, v tomto případě, zrakově handicapované uživatele webových aplikací.

V teoretické části se autor zaměří na představení základních webových technologií souvisejících s problematikou se zaměřením na JavaScript a především na samotné SpeechRecognition API a jeho implementaci a zakomponování pomocí již zmíněného JavaScriptu. V praktické části bude vytvořena webová aplikace, která bude demonstrovat možnosti, vlastnosti a výhody či nevýhody, které tato technologie rozpoznávání hlasu v současné době přináší.

Klíčová slova

HTML, CSS, JavaScript, Speech Recognition, Speech Synthesis, API, webové rozhraní, front-end, komponenta

Abstract

The point of the bachelor's thesis is to process the issue of voice recognition and voice input using the JavaScript interface SpeechRecognition. This interface is part of the Web Speech API, which allows you to integrate voice data into web applications and contains two components - text-to-speech (SpeechSynthesis) and asynchronous speech recognition (SpeechRecognition). JS API technology allows you to recognize the voice coming into the input, then convert it to text, which can then be used in many ways, such as simply typing as a dictaphone, or invoking other functions using a word command. A very positive and especially current feature is the fact that it is a so-called blind friendly technique, ie technology designed for visually impaired users of web applications in this case.

In the theoretical part, the author will focus on the introduction of basic web technologies related to the issue with a focus on JavaScript and especially on the SpeechRecognition API itself and its implementation and incorporation using the aforementioned JavaScript. In the practical part, a web application will be created, which will demonstrate the possibilities, features and advantages or disadvantages that this voice recognition technology currently brings.

Keywords

HTML, CSS, JavaScript, Speech Recognition, Speech Synthesis, API, web interface, front-end, component

Obsah

1 Úvod

1.1 Východiska práce

Webové stránky byly původně čistě statické soubory, které měly předat velké množství informací, pokud možno co největší skupině lidí. Avšak s narůstajícím výkonem strojů a rostoucími požadavky uživatelů se k webovému rozhraní dostalo i velké kvantum funkčních možností. Jednou z nich je i rozpoznávání hlasu a jeho převod na text pomocí Speech Recognition JS API. Jak už samotný název napovídá, umožňuje rozpoznávat hlas a převádět ho na text, a to díky JavaScriptovému rozhraní, které se následně postará o volání funkcí námi zadaných. Jaké funkce předložíme, je na našich potřebách, může to být ukládání poznámek či úkolů, nebo nějaká vizuální změna na stránce, možnosti jsou omezené pouze naší představivostí. Jako velkou výhodu vidím i možnosti používání webu nevidomými lidmi a obecné usnadnění práce s webem.

1.2 Cíle práce

Cílem bakalářské práce bude teoreticky popsat využití Speech Recognition JS API a samozřejmě najít jeho výhody a nevýhody. Následně v praktické části vytvořím několik jednoduchých komponent, které budou názorně ukazovat možnosti využití hlasového zadávání pomocí Speech Recognition JS API.

Od výsledku si slibuji potvrzení či vyvrácení výhod a nevýhod a také vytvoření základní domovské stránky, takzvaného rozcestníku, který bude umožňovat jednodušší navigaci na vyhledávanou stránku či demonstrativní komponenty. Vše bude uhlazeno základním stylingem pomocí čistého CSS.

1.3 Metody práce

V úvodu práce vyhraním veškeré pojmy, které budou nutné pro uchopení této problematiky. Dále se zaměřím na určení jasných výhod a nevýhod rozpoznávání hlasu uvnitř webového rozhraní. A následně se přesunu na tvorbu jednotlivých komponent, které budou demonstrovat samotné využití rozpoznávání hlasu. Z těchto komponent nakonec získám potvrzení nebo vyvrácení předem určených pozitiv či negativ.

2 Použité technologie

2.1 HTML

HTML, jinak také HyperText Markup Language, je základním stavebním kamenem pro tvorbu webu. Dále se také používá v kombinace s dalšími technologiemi, které umožňují kostru tvořenou HTML stylizovat (CSS) nebo jí přidávají funkcionální schopnosti a chování (JavaScript). [1]

Pojem „HYPERTEXT"zastupuje takzvané (hypertextové) odkazy, které umožňují propojení mezi jednotlivými stránkami v rámci jednoho webu, či mohou odkazovat na stránku jiného webu v internetu.[1]

„Markup"využívá značek, které vysvětlí prohlížeči, jak má zobrazit text, obrázky a ostatní kontent pro webový prohlížeč. HTML markup zahrnuje speciální elementy jako jsou například: [1]

```
1  <head> <body> <title> <div> <input> <img> <span> <p> <h1>
```

Příklad 1: Příklad elementů

a mnoho dalších. Každý element je uveden danou značkou, takzvaným tagem. Některé elementy používají ukončovací značku (body, div), kterou obklopují daný kontent, jiné to však nevyžadují (img, input) viz. kód „ukázka HTML". Úvodní značka může obsahovat jeden či více atributů, které umožní předat elementu hodnoty, či změnit jeho chování. [1]

2.1.1 HTML5

Termín HTML5 je v podstatě trendy způsob, jak nazývat set moderních webových technologií. HTML jako základ, společně s JavaScriptovými rozhraními, vylepšuje přístup k úložišti, multimédiím a hardwaru. Jakákoliv moderní stránka by měla používat HTML doctype, aby si zajistila, že používá nejnovější verzi HTML. Z těchto technologií stojí za zmínku například Web Storage API, Drag and Drop API nebo Geolocation API. [2][3]

- **Web storage API**

  Toto rozhraní dává prohlížeči schopnost uchovávat data. Dříve se takovýto typ dat uchovával převážně formou cookies. HTML5 však nabídlo bezpečnější způsob jak data uchovat. Zároveň může uchovat mnohem větší množství dat než cookies. [3]

- **Drag & Drop API**

  Pomocí toho rozhraní můžeme uchopit element a z jednoho místa ho přenést na jiné. Vyžaduje element, který má nastavený atribut "draggable"na hodnotu "true"a kontejner s funkcí, která mu umožní element připnout. Tato funkce využívá metody appendChild. [3]

- **Geolocation API**

  Má schopnost sdělit vaši pozici pomocí globálních souřadnic (zeměpisná šířka a délka). Souřadnice uživatele se umožní odkrýt jen za předpokladu, že to uživatel povolí, protože to ohrožuje bezpečnost a soukromí uživatele. Nejširší využití má u vývoje a práce s mapami. [3]

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Example</title>
5      </head>
6      <body>
7          <h1>Example of HTML!</h1>
8          <img alt="html" src="/img/html.jpg">
9      </body>
10 </html>
```

Příklad 2: Ukázka HTML

2.2 CSS

Zkratka CSS znamená ve své dlouhé verzi Cascading Style Sheets a je to jazyk, který slouží k úpravě webových stránek. Zároveň však umožňuje upravovat nejen HTML soubory, ale i soubory typu XML. CSS popisuje, jak by se měli elementy na obrazovce vykreslovat, a to včetně struktury, rozložení, barev, fontů písma a mnoha dalších. K elementům přistupuje buď pomocí tagů nebo tříd. Velkou výhodou CSS je, že umožňuje adaptaci stylů pro různá zařízení, jako jsou počítače, mobily či tablety. Zároveň nám dává plnou moc, i co se týče různých rozměrů a poměrů obrazovek a monitorů. Požadovaného chování dosáhneme pomocí takzvaných Media Queries, které nám umožní nastavit selektivně specifické styly, pokud je obrazovka uživatele větší než určitá hodnota, viz. ukázka CSS.[4][5]

```
1  div{
2      width: 100px;
3      height: 100px;
4  }
5
6  .className{
7      color: blue;
8  }
9
10 @media only screen and (max-width: 600px) {
11   body {
12     background-color: lightblue;
13   }
14 }
```

Příklad 3: Ukázka CSS

2.2.1 Responzivní layout technologie

Responzivní weby jsou postaveny na flexibilních mřížkách, což znamená, že není třeba cílit na každou možnou velikost zařízení pomocí rozvržení s dokonalými pixely.[5]

Pomocí flexibilní mřížky lze změnit prvek nebo přidat bod přerušení a změnit tak design v bodě, kde obsah začíná vypadat špatně nebo přetéká. Lze například zajistit, aby se délka řádků nestala nečitelnou při zvětšování velikosti obrazovky, a to za pomoci "column"neboli sloupce. Pokud se rámeček při zužování zmáčkne se dvěma slovy na každém řádku, nastaví se zarážka.[5]

Ve výchozím nastavení reaguje několik metod rozvržení, včetně rozvržení více sloupci, flexbox a grid. [5]

- **Multicol**

  Pomocí multicol se určí počet sloupců. Prohlížeč si sám zjistí jejich velikost, která se bude odvíjet od rozměru obrazovky.[5]

- **Flexbox**

  Ve flexboxu se ohebné předměty zmenšují nebo rostou a rozdělují si prostor podle rozměrů jejich kontejneru.[5]

- **CSS grid**

  V rozvržení mřížky se využívá jednotky "fr"(fraction), která umožňuje distribuci dostupného prostoru napříč drahami mřížky.[5]

2.3 JavaScript

JavaScript (JS) je skriptovací jazyk, který umožňuje implementovat komplexní funkce. Pokaždé, co se zobrazuje nějaký obsah, je veliká pravděpodobnost, že je zapojený JavaScript. Dává možnost dynamicky upravovat kontent, kontrolovat multimédia, animovat obrázky a jiné. Ve výsledku záleží na schopnostech a zkušenostech programátora. JavaSript je určen převážně na práci s klientskou stranou aplikace, ale již zmíněná APIs(rozhraní) přidávají extra super schopnosti, které lze využít v kódu. Jako jednou z možností na straně serveru se ukázal být populární Node.js. [6]

Jak už z názvu vyplývá, kód psaný v JavaScriptu se nazývá skript. A jeho části se buď pomocí script tagů vkládají na konec hlavičky v HTML, nebo mohou být přidány pomocí separátního souboru a naimportovány do HTML, a to opět na konec hlavičky. Avšak místo kódu vložíme pouze zdroj na soubor s naším skriptem. [6]

```
1 <!-- Internal JavaScript Code -->
2 <body>
3     <script>
4         console.log("Hello world");
5     </script>
6 </body>
```

Příklad 4: Ukázka interního vkládání JS kódu

```
1 <!-- External JavaScript Code -->
2 <head>
3     <script src="main.js"></script>
4 </head>
```

Příklad 5: Ukázka externího vkládání JS kódu

2.3.1 Případy použití JavaScriptu

JavaScript se využívá převážně na vývoj webových stránek a web-based aplikací. Zároveň je to ale jazyk, který může být použitý jak ve frontendu, tak v backendu. Některé případy použití JavaScriptu jsou například: [7]

- **Vytváření interaktivních webových stránek**

  Používá se pro své schopnosti dynamicky měnit jakýkoliv kontent a styly webové stránky. [7]

- **Vyvíjení aplikací**

  JavaScript se používá nejen pro vývoj webových, ale i mobilních aplikací. Některými z populárních frameworků jsou ReactJS nebo React Native.[7]

- **Web servery**

  Lze vytvořit i robustní serverové aplikace, avšak abychom mohli takovýto server postavit, vyžaduje to využití frameworků Node.js nebo Express.[7]

- **Vývoj her**

  Javascript lze využít i pro vývoj a design prohlížečových her. Nalézá se zde mnoho herních enginů, které poskytují framworky pro produkci her.[7]

```
1 function helloWorld(){
2     console.log("Hello world!");
3 }
4
5 helloWorld(); // --> obsah konzole: "Hello world!"
```

Příklad 6: Ukázka JS

2.3.2 Síla JavaScriptu

JavaScript je pokročilý programovací jazyk, který dělá webové stránky interaktivní a dynamické, zatímco HTML zprostředkovává pouze strukturu, neboli kostru stránky. Jeho hlavní silou je, že dynamicky přidává či mění kontent na stránce a díky tomu může stránka vypadat lépe a mít mnohem více schopností než statické HTML. [7]

2.3.3 Pozice JavaScriptu v dnešní době

JavaScript je jedním z nejpopulárnějších programovacích jazyků na světě. Je spousta důvodů, díky kterým je JavaScript velmi chtěný a programátor, který ho ovládá dobře, je ve velmi dobré pozici. Některé z důvodů jsou například: [7]

- **Nevyžaduje kompilátor**

  Poněvadž je JS interpretační jazyk, nevyžaduje žádné speciální kompilátory ani nemusí být pro použití předem kompilován. [7]

- **Využívá jak klientské strany tak strany serveru**

  Zpočátku byl JS využíván převážně na stavbu klientských aplikací. Postupem času však narostly jeho schopnosti a s příchodem a evolucí frameworků se nyní široce používá na stavbu serverových aplikací.[7]

- **Pomáhá vytvořit kompletní produkt**

  Jak už bylo zmíněno, JS umožňuje využití jak klientské, tak serverové strany, to z něj dělá dobrého pomocníka na tvorbu "end-to-end"produktů.

  [7]

- **Používá se všude**

  JavaScript je milován, protože může být použit kdekoliv při vývoji webových stránek, her, mobilních aplikací a mnoho dalšího.[7]

- **Obrovská komunita**

  JS má obrovskou komunitu uživatelů a mentorů. A díky tomu se stále posouvá dál. Zároveň je velmi příjemným začátkem pro každého programátora, protože díky velké komunitě vždy najde odpověď na svoje otázky.[7]

2.3.4 Představení JavaScriptu

JavaScript je jak imperativní, tak deklarativní typ jazyka. Obsahuje standartní objekty, jako spousta ostatních programovacích jazyků, jako jsou například Array(list), Date(datum) nebo Math(matematická knihovna). Také má základní soubor jazykových prvků, jako jsou operátory, kontrolní struktury a statementy.[8]

Klientská strana zprostředkovává pomocí prohlížeče takzvaný DOM (Document Object Model). Tento model umožňuje aplikaci vkládat elementy do HTML a odpovídat na eventy(události), jako jsou například kliknutí myší, formulářové vstupy nebo navigace po stránce. Pro práci s klientskou stranou vznikají velmi mocné frameworky jako jsou AngularJS, ReactJS nebo VueJS.[8]

Serverová strana umožňuje aplikaci komunikovat s databází a zprostředkovává tok informací vyvolaný jednou aplikací na další. Také může manipulovat se složkami na serveru. Nejpoužívanějším frameworkem je již zmíněný node.js.[8]

2.3.5 Limitace JavaScriptu

Bezpečností riziko: JavaScript může být použit pro takzvané "fetchování"dat například pomocí knihovny AJAX nebo manipulováním tagů, které tato data načítají, jako jsou například <img>,<object>,<script>. Tento způsob útoku se nazývá cross-site skriptový útok. JavaScript, který není původní součástí stránky, je injektován do stránky a díky tomu může dosáhnout neočekávaných situací a například poškodit stránku či ji shodit.[8]

Výkonost: JavaScript neumožňuje stejnou míru výkonu jako nabízejí jiné tradiční programovací jazyky. Avšak JS zpravidla vykonává jednoduché činnosti v rámci prohlížeče, takže výkon není považován za velké omezení při jeho používání.[8]

Složitost: Aby si programátor osvojil skriptovací jazyk, musí posbírat mnoho znalostí ohledně všech programovacích konceptů, jazykových objektů a ostatních problematik. Jinak nebude schopný psát pokročilé skripty pomocí JS.[8]

Slabé zařízení pro zpracování chyb a kontrolu typu: Je to slabě typový jazyk. Není nutné specifikovat typ proměnné, takže nedochází ke kontrole typu. Zároveň, chybové hlášky, které JS vypisuje do konzole, ne vždy poradí a pomohou s řešením problému.[8]

2.4 API

API, jinak také aplikační programové rozhraní, je sada definovaných pravidel, která umožňují různým aplikacím, aby spolu komunikovaly. Funguje jako zprostředkovatelská vrstva, která zpracovává datové přenosy mezi systémy a umožňuje společnostem otevřít svá aplikační data a funkce externím vývojářům třetích stran, či obchodním partnerům a interním oddělením v rámci společnosti.[9]

Definice a protokoly, které jsou v rámci API, pomáhají společnostem a podnikům propojit mnoho různých aplikací, které používají v každodenních operacích. To šetří čas zaměstnancům a odbourává překážky, které mohou bránit spolupráci a inovaci. Pro vývojáře poskytuje dokumentace API možnosti a rozhraní pro komunikaci aplikací. To následně zjednodušuje integraci aplikací.[9]

2.4.1 Funkce API

Nejjednodušším způsobem, jak pochopit fungování rozhraní API, je podívat se na příklad z praxe. Takovým příkladem může být třeba zpracování plateb třetí stranou. Pokud uživatel zakoupí produkt přes webovou stránku elektronického obchodu, může být vyzván k zaplacení například přes Paypal nebo jiný tomu podobný systém třetí strany. Tato funkce se při navazování spojení vždy spoléhá na rozhraní API. I přesto, že přenos dat se bude lišit v závislosti na používané webové službě, všechny požadavky a odpovědi probíhají prostřednictvím rozhraní API. Uživatelské rozhraní na tyto požadavky nevidí, což znamená, že rozhraní si vyměňují data v počítači nebo aplikaci a uživateli se to jeví jako bezproblémové připojení.[9]

Příklad navázání spojení lze rozdělit do bodového postupu. [9]

- Když kupující klikne na tlačítko platby, API začne volání k načtení informací. Tomu se také jinak říká požadavek. Tento požadavek je zpracován z aplikace směrem na webový server, a to pomocí rozhraní URI (Uniform Resource Identifier) a zahrnuje takzvaný slovesný požadavek (GET, PUT, POST apod.), záhlaví a někdy i tělo požadavku dle specifikací.[9]
- Po obdržení platného požadavku z webové stránky produktu, zavolá API externí program nebo webový server, v tomto případě je to platební systém třetí strany.[9]
- Server následně odešle odpověď API s požadovanými informacemi.[9]
- Rozhraní API přenese data zpět do počáteční aplikace, která si o ně zažádala. V tomto případě na webové stránky produktu.[9]

2.4.2 Výhody API

Rozhraní API zjednodušují návrh a vývoj nových aplikací, služeb a integraci a správu těch stávajících. Nabízejí však i další velmi významné výhody vývojářům a organizacím obecně.[9]

Vylepšená spolupráce: Průměrný podnik dnešní doby, používá téměř 1200 cloudových aplikací. Rozhraní API umožňují integraci, takže tyto platformy a aplikace mohou mezi sebou bez problémů komunikovat. Za pomoci této integrace mohou společnosti automatizovat pracovní postupy a zlepšit tak spolupráci jednotlivých pracovišť. Bez API by mnoho podniků postrádalo konektivitu, což by způsobilo informační překážky a ty by ohrozily produktivitu a výkon společnosti.[9]

Zrychlená inovace: Rozhraní API nabízejí flexibilitu a dávají společnostem možnost navazovat spojení s novými obchodními partnery, nabízet nové služby na jejich stávajícím trhu a v konečném důsledku přistupovat na trhy

nové, které následně mohou vytvářet masivní výnosy a řídit digitální transformaci. Pro příklad společnost Stripe začínala jako API s pouhými sedmi řádky kódu. Tato společnost od té doby uzavřela partnerství s mnoha velkými podniky ve světě, následně se diverzifikovala tak, že poskytovala půjčky a firemní karty, a nedávno byla oceněna na 36 miliard dolarů.[9]

Zpeněžení dat: Mnoho společností se rozhodlo nabízet svá API zdarma, alespoň zpočátku, aby mohly kolem své značky vybudovat publikum vývojářů a navázat tak vztahy s potencionálními obchodními partnery. Pokud API poskytuje přístup k cenným digitálním aktivům, firma je může zpeněžit prodejem přístupu. Toto se nazývá ekonomika API. Příkladem může být AccuWeather. Ten spustil svůj samoobslužný vývojářský portál k prodeji široké škály balíčků API. Trvalo to krátkých 10 měsíců, než přilákal 24 000 vývojářů, prodal 11 000 klíčů API a během tohoto procesu vybudoval prosperující a rostoucí komunitu.[9]

Zabezpečení systému: Rozhraní API oddělují požadující aplikaci od infrastruktury odpovídající služby a při komunikaci mezi nimi nabízejí vrstvy, které přidají extra zabezpečení. Například volání API obvykle vyžaduje autentizační pověření. HTTP hlavičky, soubory cookie nebo řetězce dotazů umožňují poskytnout další zabezpečení během přenosu dat a brána API může řídit pouze přístup, aby se dále minimalizovaly bezpečností hrozby a úniky. [9]

Zabezpečení a soukromí koncového uživatele: Tak jako API poskytují přidanou ochranu v rámci sítě, umožňují také poskytovat další vrstvu ochrany pro koncového uživatele. Některé weby požadují polohu uživatele, která je poskytována prostřednictvím rozhraní API (Geolocation API). Uživatel má následně na výběr, zdali tento požadavek povolí, či se ho rozhodne zamítnout. Mnoho webových prohlížečů a mobilních operačních systémů, jako je například iOS, má vestavěné struktury oprávnění, které se užívají, když rozhraní API požaduje přístup k aplikacím a jejich datům. Pokud musí aplikace přistupovat k souborům prostřednictvím rozhraní API, souborové systémy, jako Windows, Mac a Linux, používají oprávnění pro tento přístup. [9]

2.4.3 Běžné užití API

Díky tomu, že API umožňují společnostem otevřít přístup ke svým zdrojům, ale zároveň zachovat bezpečnost a kontrolu, staly se velmi cenným aspektem moderních obchodních a osobních aplikací. Zde je několik běžných případů použití API z praxe, se kterými se uživatel může setkat téměř každý den. [9]

Univerzální přihlášení: Jedním z populárních příkladů rozhraní API je funkce, která lidem umožňuje přihlásit se na stránky pomocí přihlašovacích údajů jiných profilů, jakou jsou například Facebook, Twitter nebo Google. Tato pohodlná funkce umožňuje libovolné webové stránce využít API z jedné z nejpopulárnějších služeb pro rychlou autentizaci, což šetří čas a námahu s nastavováním nového profilu pro každou webovou aplikaci nebo členství. [9]

IoT - Internet of Things: Pod překladem internet věcí se nedá moc představit, každopádně tato chytrá zařízení nabízejí další funkce, jako jsou dotykové obrazovky s internetem a sběr dat pomocí rozhraní API. Chytrá lednice se může například připojit k aplikacím s recepty nebo si psát a odesílat poznámky do mobilních telefonů prostřednictvím textové zprávy. Vnitřní kamery se připojují k různým aplikacím, takže uživatel může vidět obsah chladničky například při nakupování. [9]

Srovnávání zájezdů: Stránky, které umožňují rezervace zájezdů, shromažďují tisíce letů a představují nejlevnější možnosti pro každé datum a destinaci. Tato služba je umožněna prostřednictvím API a uživatelům aplikace poskytuje přístup k nejnovějším informacím o dostupnosti hotelů a leteckých společností, a to buď prostřednictvím webového prohlížeče, nebo vlastní mobilní aplikace. Díky autonomní výměně dat a požadavků se API dramaticky zkracuje čas a úsilí spojené s kontrolou dostupnosti letů a ubytování. [9]

Mapovací aplikace: Pokud pomineme základní rozhraní API, která zobrazují statické nebo interaktivní mapy, tak tyto aplikace používají spousty dalších rozhraní a funkcí, které poskytují uživatelům trasy, rychlostní limity, dopravní varování, body zájmů a další. Uživatelé komunikují s rozhraním API

při vykreslování cestovních tras nebo při sledování předmětů na cestách, jako je dodávkové vozidlo nějaké dopravní společnosti. [9]

Twitter: Každý tweet, neboli zpráva na twitteru, obsahuje základní atributy, včetně autora, jedinečného ID zprávy, časového razítka, respektive kdy byl zveřejněn, a geolokační metadata. Twitter zpřístupňuje základní atributy veřejných tweetů a odpovědí vývojářům a umožňuje jim zveřejňovat tweety na jiných webových stránkách prostřednictvím rozhraní API společnosti. [9]

SaaS aplikace: Rozhraní API jsou nedílnou součástí SaaS neboli "Software-as-a-service". Platformy jako CRM (nástroj pro správu vztahů se zákazníky) často obsahují řadu vestavěných rozhraní API, která společnostem umožňují integraci s aplikacemi, které již používají, jako jsou zasílání zpráv, sociální média a emailové aplikace. To dramaticky snižuje čas strávený při přepínání mezi aplikacemi pro prodejní a marketingové účely.[9]

2.4.4 Typy API rozhraní

V dnešní době je většina API rozhraními webovými, která zpřístupňují data a funkce aplikace přes internet. Zde jsou čtyři hlavní typy webových rozhraní API: [9]

- **Open API**

  V překladu otevřená rozhraní API jsou rozhraní pro programování aplikací s otevřeným zdrojovým kódem, ke kterým máte přístup pomocí protokolu HTTP. Jsou také známá jako veřejná rozhraní, mají definované koncové body API a formáty požadavků a odpovědí. [9]

- **Partner API**

  Jsou partnerská API, která propojují strategické obchodní partnery. Vývojáři k těmto rozhraním API obvykle přistupují v samoobslužném režimu prostřednictvím veřejného portálu pro vývojáře. Přesto musí dokončit proces registrace a získat tak přihlašovací údaje pro přístup k partnerskému rozhraní API. [9]

- **Internal API**

  Jsou interní rozhraní, která zůstávají externím uživatelům skryta. Tato soukromá rozhraní API nejsou dostupná pro uživatele mimo společnost a jsou určena ke zlepšení produktivity a komunikace mezi různými interními vývojovými týmy. [9]

- **Composite API**

  Neboli složená rozhraní API kombinují více datových nebo servisních rozhraní. Umožňují tak přístup k více koncovým bodům v jednom. [9]

2.5 Speech recognition

Rozpoznávání řeči, známe také jako automatické rozpoznávání řeči (ASR), počítačové rozpoznávání řeči nebo převod řeči na text. Je to schopnost, která umožňuje programu odchytávat a zpracovávat lidskou řeč do písemného neboli textového formátu. Dost často se zaměňuje s rozpoznáním hlasu. Rozpoznání řeči se však zaměřuje na překlad řeči z verbálního formátu na formát textový, zatímco rozpoznání hlasu se snaží o identifikaci hlasu jednotlivého uživatele.[10]

2.5.1 Klíčové vlastnosti efektivního rozpoznávání řeči

K dispozici jsou mnohé aplikace a zařízení pro rozpoznávání řeči, ale nejpokročilejším řešením je umělá inteligence a strojové učení. To totiž integruje gramatiku, syntax, strukturu a složení zvukových a hlasových signálů pro porozumění a zpracování lidské řeči. V ideálním případě se učí za pochodu, což znamená, že se vyvíjí reakce s každou interakcí. [10]

Nejlepší druh systémů také umožňuje organizacím přizpůsobit a adaptovat technologii jejich specifickým požadavkům. Vše od jazyka a nuancí řeči, až po rozpoznání značky. Některé z příkladů jsou: [10]

- **Vážení jazyka**

  Zvyšuje přesnost vážení konkrétních slov, která se často používají (jako jsou názvy produktů nebo oborový slang), nad rámec výrazů, které jsou již v základní slovní zásobě. [10]

- **Označení řečníka**

  Vytváří přepis, který cituje nebo označí příspěvky každého řečníka z konverzace s více účastníky. [10]

- **Akustický trénink**

  Věnuje se akustické stránce podniku. Naučí systém, aby se přizpůsobil akustickému prostředí (jako je okolní hluk v call centru) a stylům reproduktorů, což zahrnuje výšku hlasu, hlasitost a tempo.[10]

- **Filtrace vulgárních slov**

  Používá filtry k identifikaci určitých slov nebo frází a následně vytvoří čistý, vycenzurovaný řečový výstup. [10]

2.5.2 Případy užití Speech recognition

Různé aplikace řečových technologií dnes využívá velká řada průmyslových odvětví. Tyto aplikace pomáhají podnikům i spotřebitelům šetřit čas, a dokonce i životy. Některé příklady zahrnují: [10]

- **Automobilový průmysl**

  Rozpoznávače řeči zvyšují bezpečnost řidičů tím, že umožňují hlasové navigování po vnitřních systémech auta či vyhledávání v autorádiích.[10]

- **Technologie**

  Virtuální agenti se stále více integrují do našeho každodenního života, zejména do našich mobilních zařízení. Používáme k nim hlasové příkazy prostřednictvím našich chytrých telefonů, například přes Google Assistant nebo přes Apple Siri, pro úkoly, jako je například hlasové vyhledávání. Také existují inteligentní reproduktory, jako například Amazon Alexa nebo Microsoft Cortana pro přehrávání hudby. Jejich integrace do každodenních produktů, které používáme, bude jen pokračovat, což podpoří hnutí "internetu věcí". [10]

- **Zdravotní péče**

  Lékaři a sestry využívají diktafonové aplikace k zachycení a protokolování diagnóz pacientů a poznámek k léčbě. [10]

- **Prodeje**

  Technologie rozpoznávání řeči má v prodeji několik využití. Call centru může pomoci přepsat tisíce telefonních hovorů mezi zákazníky a agenty, aby mohlo dojít k identifikaci běžného vzorce hovorů a vznikajících problémů. Chatboti AI mohou také mluvit s lidmi prostřednictvím webové stránky, odpovídat na běžné otázky a řešit základní požadavky zákazníka, aniž by musel čekat až bude k dispozici agent kontaktního centra. V obou případech systémy rozpoznávání řeči pomáhají zkrátit dobu potřebnou k vyřešení spotřebitelských problémů. [10]

- **Zabezpečení**

  Během toho, co se technologie integrují do našeho každodenního života, bezpečnostní protokoly jsou stále důležitější. Hlasová autentizace přidává životaschopnou úroveň zabezpečení.[10]

2.5.3 Webové speech API

Web Speech API umožňuje začlenit hlasová data do webových aplikací. Web speech API má dvě části, SpeechRecognition, což je asynchronní rozpoznávání řeči a převod na text a SpeechSynthesis, které je pravým opakem a převádí text na řeč. [11]

2.5.4 SpeechRecognition

K rozpoznávání řeči se přistupuje prostřednictvím rozhraní SpeechRecognition, které umožňuje rozpoznat hlasový kontext ze zvukového vstupu (obvykle prostřednictvím výchozí služby zařízení) a vhodně reagovat. Obecně se využívá

konstruktoru k vytvoření nového objektu SpeechRecognition, který má k dispozici řadu obslužných rutin, respektive událostí pro detekci vstupu řeči přes mikrofon zařízení. SpeechRecognition také dědí vlastnosti svého nadřazeného rozhraní EventTarget. Mezi tyto vlastnosti patří: [11][12]

- **SpeechRecognition.grammars**

  Lze uzpůsobit gramatiku, které bude aktuální rozpoznávání řeči rozumět.[12]

- **SpeechRecognition.lang**

  Umožňuje přizpůsobit jazyk, kterému bude rozpoznání rozumět. Pokud není zadáno, použije se výchozí hodnota atributu jazyka HTML nebo jazykového nastavení uživatelského agenta.[12]

- **SpeechRecognition.continuous**

  Tato vlastnost řídí, zda se pro každé rozpoznání vrátí průběžný výsledek, nebo pouze jeden výsledek. [12]

- **SpeechRecognition.interimResults**

  Řídí, zdali mají být vráceny průběžné výsledky nebo ne. Průběžné výsledky jsou výsledky, které ještě nejsou konečné.[12]

- **SpeechRecognition.maxAlternatives**

  Nastaví maximální počet alternativ poskytnutých na výsledek. Výchozí hodnota je 1.[12]

2.5.5 SpeechSynthesis

K syntéze řeči lze přistupovat prostřednictvím rozhraní SpeechSynthesis, což je součást převodu textu na řeč, která umožňuje programům číst jejich textový obsah (obvykle prostřednictvím výchozího syntetizéru zařízení). Různé typy hlasu jsou prezentovány objekty SpeechSynthessVoice a různými částmi textu, které mají být mluvené. Vlastnosti SpeechSynthesis jsou: [11]

- **SpeechSynthesis.paused**

  Booleovská hodnota, která je pravdivá, pokud je object SpeechSynthesis v pozastaveném režimu.[13]

- **SpeechSynthesis.pending**

  Booleovská hodnota, která je pravdivá, pokud fronta promluvy obsahuje doposud nevyslovené projevy.[13]

- **SpeechSynthesis.speaking**

  Booleovská hodnota, která je pravdivá, pokud právě probíhá vyslovování výroku, a to i za předpokladu, kdy je SpeechSynthesis pozastaveno.[13]

2.5.6 Podpora Web Speech API

SpeechRecognition je v současné době omezeno na Chrome, a to jak pro verzi počítačovou, tak pro Android verzi. Chrome podporuje toto rozhraní přibližně od verze 33. Zároveň je ale třeba zahrnout i verzi s prefixem:[14]

```
1 const SpeechRecognition = window.SpeechRecognition ||
    webkitSpeechRecognition;
```

Příklad 7: Deklarace SpeechRecognition

Z druhé stránky podpora SpeechSynthesis se stále rozšiřuje do všech běžných prohlížečů a v současnosti je omezena na následující:[14]

- Firefox pro stolní počítače a mobilní zařízení je podporován bez prefixů. Lze jej zapnout pomocí nastavení media.webspeech.synth.enable na hodnotu true v about:config.[14]

- Firefox OS 2.5+ jej podporuje ve výchozím nastavení, a to bez nutnosti přidání jakýchkoli prefixů.[14]

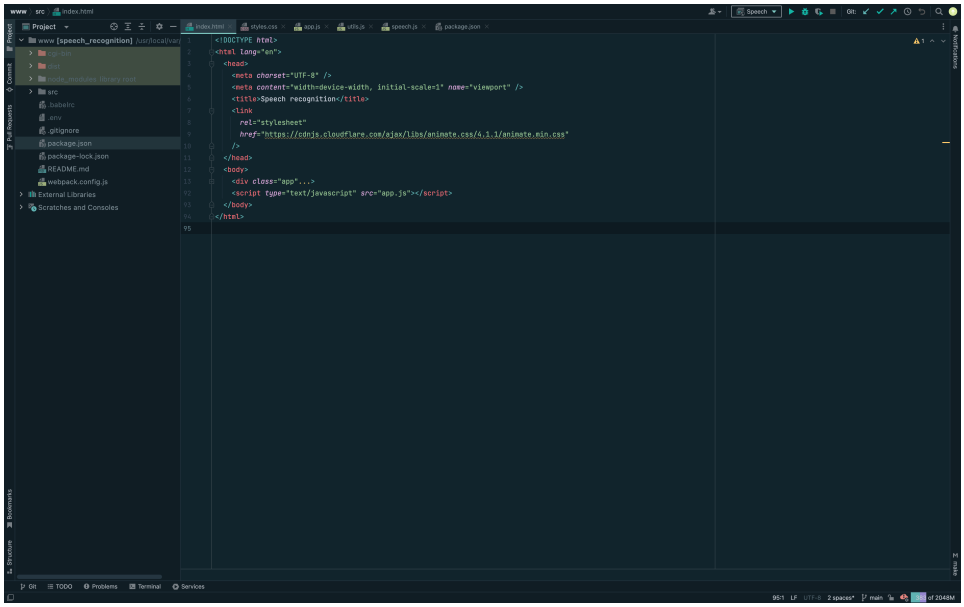
- Chrome pro stolní počítače a Android jej podporuje od verze 33 bez předpon.[14]

3 Vývojové prostředí

IDE neboli integrované vývojové prostředí umožňuje programátorům konsolidovat různé aspekty psaní programu, aplikace či webové stránky.[15]

Takové prostředí zvyšuje produktivitu programátorů spojením běžných činností psaní softwaru do jediné aplikace: vytváření spustitelných souborů, úpravy zdrojového kódu nebo ladění. Psaní kódu je důležitou součástí programování. Na začátku je prázdný soubor, po napsaní pár řádků je program na světě. IDE tento proces usnadňuje funkcemi, jako je zvýrazňování syntaxe a automatické doplňování. [15]

Mezi nejpopulárnější IDE patří Visual Studio nebo mnou zvolený WebStorm od společnosti JetBrains.



Obrázek 1: Ukázka rozhraní WebStorm

4 Výhody a nevýhody SpeechRecognition

V této části práce popíšu výhody a nevýhody, které s sebou přináší rozhraní SpeechRecognition.

4.1 Výhody

Mezi hlavní výhody tohoto rozhraní patří ta skutečnost, že se jedná o takzvaně blind friendly techniku. Tudíž toto rozhraní umožňuje nevidomým lidem využívat funkce webového prohlížeče. Obecně to však zvyšuje přístupnost webu pro osoby s postižením, které mohou mít potíže s psaním či navigací pomocí myši.

Velkou výhodou je také rychlejší zadávání. Díky rozpoznávání řeči může uživatel zadávat text mnohem rychleji, než by mohl psaním na klávesnici, což může zlepšit jeho produktivitu. Zároveň je toho všeho schopný bez použití rukou, což může být užitečné v situacích, kdy jsou ruce obsazené, nebo by bylo nebezpečné je použít.

Další významnou výhodou je podpora vícejazyčnosti. Rozhraní dokáže rozpoznat více jazyků, tudíž lze využít v mnoha zemích a regionech. Obecně se dá říct, že rozpoznávání řeči zlepšuje uživatelský zážitek z webu.

4.2 Nevýhody

Jako hlavní nevýhodu vidím hlavně omezenou přesnost. I když se rozpoznávání řeči každým dnem zlepšuje, stále existuje riziko chyb a nesprávné interpretace řeči. To může vézt k chybám v kódu stránky nebo například k nesprávnému vyhledávání. Je tedy na místě, aby docházelo k trénování tohoto rozhraní.

Další nevýhodou je potřeba spolehlivého internetového připojení. Když dojde k přerušení spojení, či jen narušení stability, může dojít k problémům s rozpoznáním řeči.

Na posledním místě je třeba zmínit problém se soukromím a bezpečností.

Používání rozpoznávání řeči totiž může znamenat, že jsou sbírána a ukládána určitě hlasová data. To může být problematické právě z hlediska soukromí a bezpečnosti našich dat.

4.3 Shrnutí

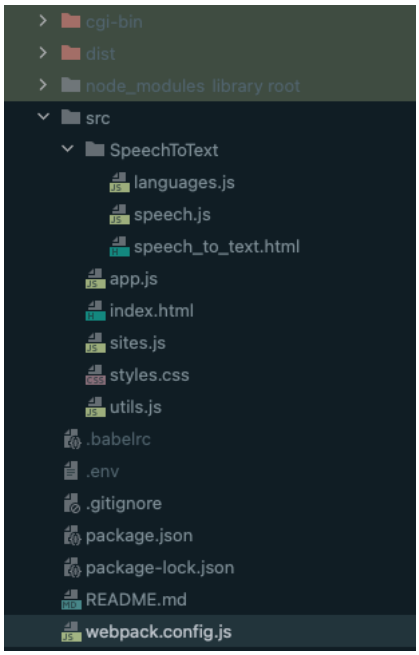
Závěrem této kapitoly bych rád zmínil, že mě překvapila moc tohoto rozhraní a také jeho schopnost s lehkostí rozpoznávat složitá česká slova. Velká slabost tohoto rozhraní tkví v tom, že jakmile je na místě nějaký ruch, rozhraní má s odchytem slov velké problémy. Pokud se tedy uživatel nachází na rušném místě, nemusí být tak úspěšný. To samé platí pro hudbu, bohužel okolní zvuky a melodie negativně narušují rozpoznání slov, takže to bohužel nefunguje jako šikovný přepisovač textu písně.

5 Praktická část

V praktické části mé bakalářské práce jsem vytvořil webovou aplikaci, která mi umožní demonstrovat sílu webového rozhraní Speech. Pro ukázku schopností tohoto balíčku, jsem vytvořil dvě komponenty. Tou hlavní je hlasově ovládaný rozcestník, který umožňuje například otevřít stránky, ale ukáže i počasí či aktuální cenu bitcoinu. Přes tento rozcestník se pomocí příkazu následně lze dostat na druhou demonstrativní komponentu, kterou je vícejazyčný převaděč hlasu na text, který v sobě má i další funkce.

5.1 Struktura projektu

Ve svém pracovním adresáři BP jsem pro větší přehlednost rozdělil soubory týkající se rozcestníku a diktafonu do separátních složek. Umožnilo mi to tak držet větší kontrolu nad JavaScriptovými soubory. Vše, týkající se rozcestníku, leží jakožto main rozhraní ve složce /src, diktafon má následně složku svojí. Viz. screenshot.



Obrázek 2: Ukázka adresářové struktury

5.2 Jednotlivé soubory

V adresáři lze vidět spoustu souborů, než se ale pustím do podrobného zkoumání, je třeba vysvětlit, co který soubor zastupuje.

5.2.1 HTML

Pro každou z demonstrativních komponent jsem vytvořil HTML soubor. Pro rozcestník jsem použil defaultní název index.html a diktafon jsem se rozhodl pojmenovat speech_to_text.html. V těchto souborech se nachází kostra celé aplikace.

5.2.2 CSS

Co se stylů týče, rozhodl jsem se, že aplikace není až zas tak rozměrná, takže jsem se snažil vytvořit co nejvíc utilitní systém class, abych mohl následně styly recyklovat pro celou aplikaci a nasázet je do jednoho souboru s generickým názvem styles.css.

5.2.3 Konfigurační soubory

Pro správné fungování projektu, bylo třeba vytvořit několik konfiguračních souborů. Tím hlavním je package.json, který umožňuje přednastavit si určité skripty, či udržuje tabulku závislostí pro běh projektu a pro jeho vývoj. Od počátku jsem si chtěl projekt verzovat a na to je nutné použít konfigurační soubor .gitignore, který umožňuje vynechat určitě složky či soubory z git repositáře, čímž se zbytečně nezahlcuje a práce s daty v repositáři je mnohem snazší. Hlavním adeptem na místo v gitignore souboru je složka node_modules, díky které mám v projektu přístupné další knihovny a moduly instalované pomocí npm. Tato složka je opravdu masivní, proto je třeba z gitu vynechat. Následně jsem zakomponoval službu webpack, která mi umožnila vytvořit build všech souborů. Tuto službu zastupuje konfigurační soubor webpack.config.js. Posledním konfiguračním souborem, který jsem do projektu zakomponoval, je .env. Tento

soubor drží informaci o environmentální promněnné, která by neměla být pro normálního uživatele dosažitelná. Touto proměnou je API klíč pro rozhraní s počasím.

5.2.4 JavaScript

Nejhojnější výskyt má v adresáři soubor typu js. Jako první jsem vytvořil JavaScriptové soubory, které tvoří funkcionalitu mých komponent. Následně jsem si vytvořil pomocný soubor utils.js, do kterého jsem si začal odkládat pomocné funkce, které nebylo třeba mít v rámci hlavních souborů a zbytečně by tvořily zmatek v kódu. Zároveň jsem si vytvořil dva pomocné soubory, které pouze drží konstanty s daty. Toto jsem udělal opět kvůli přehlednosti hlavních souborů. Tyto soubory drží defaultní stránky pro příkaz open a celý set jazyků pro diktafon.

5.3 Verzování

Jak už jsem zmínil, projekt jsem si chtěl od počátku verzovat, protože mi to zaprvé dává klid v tom, že moje práce je zálohována a zadruhé mám přehled o tom, kdy a kde se co změnilo, což mi dává větší moc nad vývojem mojí aplikace. Během vývoje jsem jednotlivé funkcionality přidával do repositáře zvlášť a díky tomu jsem získal krásnou mapu celého průběhu stavby této aplikace. V průběhu práce jsem se rozhodl vytvořit si zvláštní větev, ve které jsem následně testoval nové funkcionality a službu webpack. Síla této větve je v tom, že pokud bych se nakonec rozhodl webpack nepoužít, bude pro mě velmi snadné navrátit se do původní verze projektu. S touto službou jsem byl nakonec spokojený, tak jsem celý vývoj této větve napojil do původní verze. Průběh celého vývoje lze vidět na následující stránce na screenshotu z mého oblíbeného softwaru pro práci s gitem, kterým je GitKraken.

Obrázek 3: Mapa vývoje

5.4 Kostra aplikace

V této části popíšu základní strukturu obou demonstrativních komponent. Obě komponenty mají podobnou hlavičku(head), ve které se nachází důležitá meta data, jako je například správné znakové kódování, nebo import pomocné css knihovny, která mi umožňuje využít velké spousty animací. Kvůli podobnosti hlaviček zde ukážu pouze jednu z nich, jediným rozdílem je ve finále title stránky.

```
1 <head>
2     <meta charset="UTF-8" />
3     <meta content="width=device-width, initial-scale=1"
          name="viewport" />
4     <title>Speech recognition</title>
5     <link
6       rel="stylesheet"
7       href="https://cdnjs.cloudflare.com/ajax/libs/
            animate.css/4.1.1/animate.min.css"
8     />
9 </head>
```

Příklad 8: Hlavička rozcestníku

5.4.1 Rozcestník

Rozcestník jsem se rozhodl rozdělit do tří jednoduchých komponent, které nyní po částech popíšu. Jako první ale ukážu, jak vlastně výsledný rozcestník vypadá, aby bylo jasné, o čem zde píšu. První screenshot bude obsahovat hlavní dvě komponenty, které jsou vidět při načtení aplikace. Ty následně popíšu, a budu pokračovat na další screenshoty, ve kterých bude třetí komponenta, kterou je kontejner, který funguje jako overlay pro potřebná vyskakovací okna.

Obrázek 4: Ukázka rozcestníku

Jako první komponentu jsem vytvořil vrchní box, který obsahuje pouze nadpisek, tlačítko spouštějící hlasové zadávání a pole, ve kterém se zobrazují námi vyřčená slova. HTML struktura není nijak náročná, protože neobsahuje velké množství elementů. Avšak je hlavním aktérem tohoto rozcestníku, protože umožňuje spustit odposlech, ukázat slova a díky ní následují další funkce.

```
<div class="main box">
    <h1 class="main_title">Speech recognition API</h1>
    <button id="start" class="button">Voice command</button>
    <div class="texts box"></div>
</div>
```

Příklad 9: Hlavní box

Pro výpis jsem vytvořil druhou komponentu, spodní box, která opět obsahuje pouze nadpisek, plochu pro výpis a speciální spinner, který se odkryje, pokud dochází k natahování dat. Defaultně je tento spinner skrytý, což jsem zajistil atributem hidden

```
<div class="info box">
    <h6 class="box_title position_top">Output info</h6>
    <div hidden id="spinner" class="spinner"></div>
    <div class="output"></div>
</div>
```

Příklad 10: Výpisový box

Pro práci s rozcestníkem jsem na spodní část aplikace přidal dvě tlačítka, která usnadňují či rozšiřují schopnosti rozcestníku. Obě využívají již zmíněné třetí komponenty, kterou je overlay.

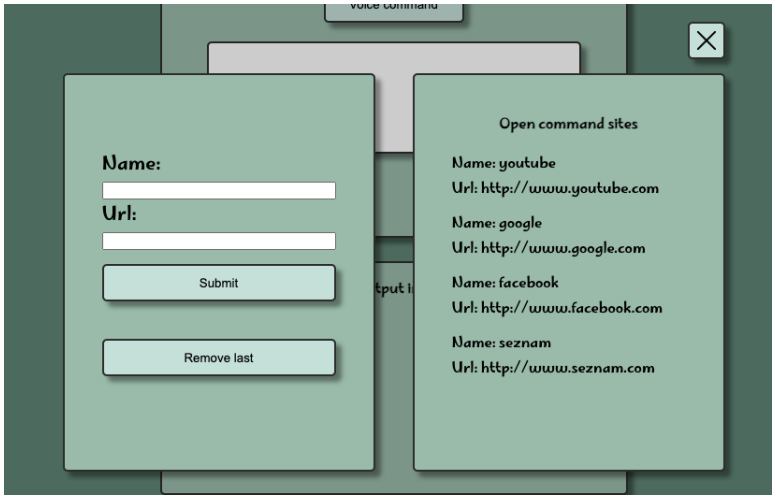
```
<button id="help" class="button">Help</button>
<button id="new" class="button">New open command</button>
```

Příklad 11: Pomocná tlačítka

Jako první jsem vytvořil tlačítko help, které otevře okno s příkazy, kterými disponuje rozcestník.

Obrázek 5: Okno s příkazy

Druhé tlačítko jsem vytvořil za účelem managementu listu stránek. Toto okno umožňuje uživateli přidávat či mazat stránky, které umí rozcestník otevřít.



Obrázek 6: Editor listu stránek

Pro práci s těmito okny, jsem vytvořil třetí komponentu. Tato komponenta se objevuje jen za předpokladu, že je potřeba. Její součástí jsou již zmíněné funkcionality tlačítek. Zároveň do ní vykresluji jednu z funkcí rozcestníku, který po vyřčení příkazu "show me my location"otevře mapu s kurentní pozicí uživatele. HTML struktura třetí overlay komponenty je mnohem komplexnější, proto jednotlivé mini komponenty vypíšu zvlášť.

```
<div class="help_box box">
    <h2>Commands</h2>
    <ul>
        <li>open + (site name)</li>
        <li>open speech to text</li>
        <li>show me my location</li>
        <li>what is the weather</li>
        <li>bitcoin price now</li>
        <li>what is the date today</li>
        <li>search + (phrase)</li>
    </ul>
</div>
```

Příklad 12: Okno s příkazy

```
<div class="site_add_container">
    <div class="site_add_box box">
        <form id="new_site" class="form">
            <label for="name" class="add_label">Name:
            </label>
            <input id="name" class="input" type="text"
               required />

            <label for="url" class="add_label">Url:
            </label>
            <input id="url" class="input" type="url"
               required />
            <button class="button submit">Submit</button>
```

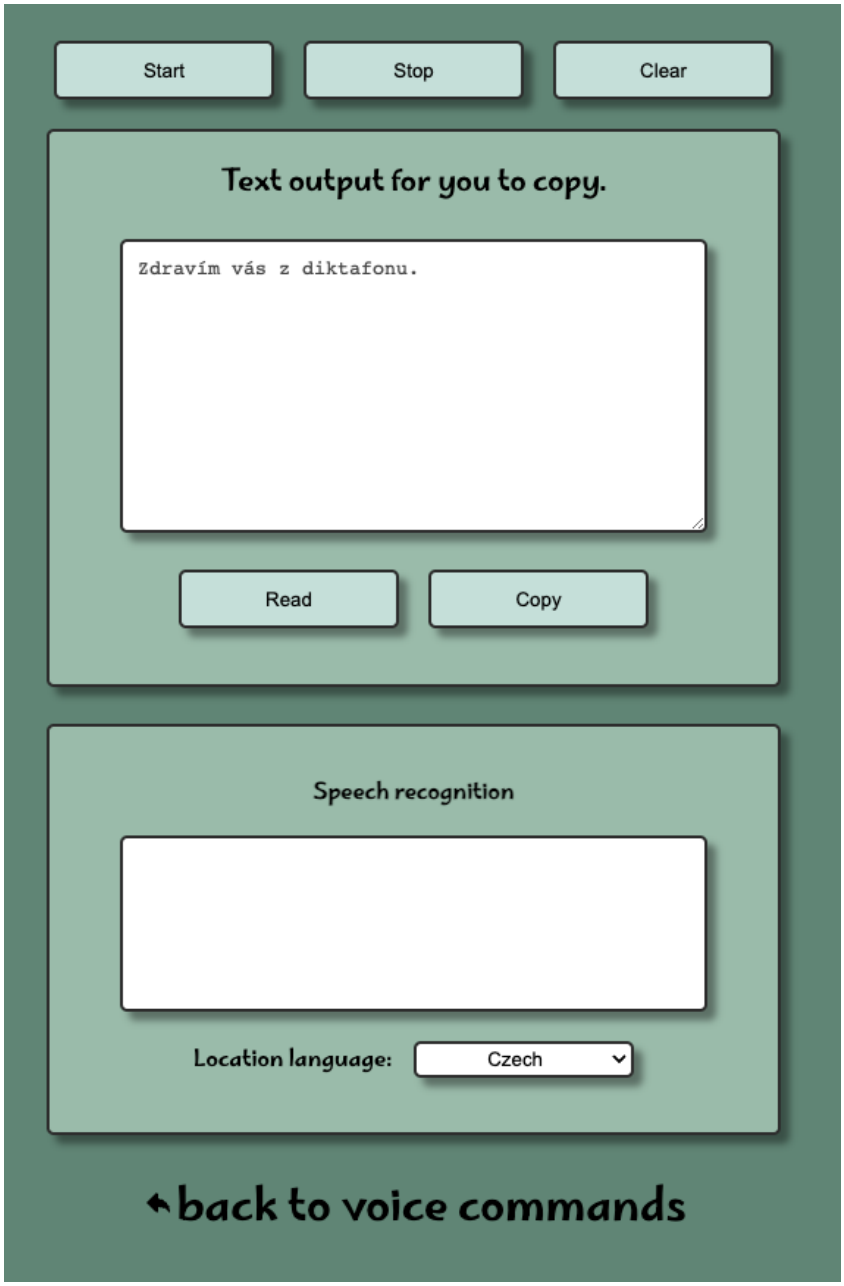
```
12              <button id="remove" class="button remove">
                    Remove last</button>
13          </form>
14      </div>
15
16      <div class="list_box_container box">
17          <h6 class="box_title">Open command sites</h6>
18          <div id="list_box" class="list_box"></div>
19      </div>
20  </div>
```

Příklad 13: Editor listu stránek

Pro vykreslení mapy se zde nachází pouze prázdný div, do kterého pomocí kódu umístím iframe s odkazem na google maps. Jako poslední věc jsem do komponenty umístil tlačítko, které mi tato vyskakovací okna umožní zavřít.

5.4.2 Diktafon

Jako druhou demonstrativní část aplikace jsem se rozhodl vytvořit více jazyčný diktafon. Ten umožňuje hlasově zadávat v mnoha světových jazycích, a následně si lze tento text nechat zpětně přečíst či okopírovat. Diktafon lze rozdělit na dvě hlavní komponenty, jedna ukazuje uživateli text, které mu rozhraní rozumělo, druhá tento text skládá za sebe, pro následné čtení či kopírování. Tyto komponenty opět demonstruji pomocí screenshotu aplikace, následně podrobněji popíšu, co je jejich úkolem a v závěru vložím kód pro ukázku.



Obrázek 7: Ukázka diktafonu

Jako první si lze povšimnout vrchních tří tlačítek. Tato tlačítka jsem se rozhodl nezahrnout do komponenty, protože jsou to hlavní ovládací tlačítka a tímto způsobem jsem dosáhl větší přehlednosti v aplikaci. První tlačítko "start"spustí hlasové zadávání, druhé tlačítko "stop"jej ukončí a třetí tlačítko "clear"vyčistí pole s textem.

```
<button id="start" class="button">Start</button>
<button id="stop" class="button">Stop</button>
<button id="clear" class="button">Clear</button>
```

Příklad 14: Tlačítka pro hlavní práci s diktafonem

První jsem vytvořil komponentu, do které se uživateli skládá text pro následnou práci. Tato komponenta obsahuje nadpisek, plochu pro text a další dvě tlačítka, která umožňují další práci s textem. První tlačítko je "read". Toto tlačítko zastupuje funkci pro zpětné čtení textu. Druhé tlačítko je "copy"a to umožňuje kopírování textu.

```
<div class="textarea_container box">
    <label class="output_heading" for="textarea">
    Text output for you to copy.
    </label>
    <textarea id="textarea" class="box textarea">
    </textarea>
    <div class="button_box">
        <button id="read" class="button">Read</button>
        <button id="copy" class="button">Copy</button>
    </div>
</div>
```

Příklad 15: První komponent diktafonu

Jako druhou komponentu jsem vytvořil box, který ukazuje slova, věty či fráze, které rozhraní rozklíčovalo z hlasu uživatele. V této komponentě se opět nachází nadpisek, plocha zobrazující aktuálně slyšený text a také menu, ve kterém si lze zvolit, který z mnoha jazyků chceme použít. V základu se nám do tohoto menu nastaví jazyk, který má uživatel zvolený jako defaultní.

```
<div class="box texts_container">
    <h6 class="box_title">Speech recognition</h6>
    <div class="texts box"></div>
    <div>
```

```
5           <label for="lang" class="label">Location language:
6           </label>
7           <select id="lang" class="dropdown"></select>
8       </div>
9   </div>
```

Příklad 16: Druhá komponent diktafonu

Na samém konci diktafonu jsem přidal link, který odkazuje zpět do hlasového rozcestníku.

5.4.3 Design

Co se týče designu, rozhodl jsem se moc si to nekomplikovat. Proto jsem vytvořil pouze jednoduchý layout, který jsem obohatil o pastelově zelenou barevnou sestavu. Všechny části aplikace jsem vylepšil pomocí stínů a okrajů. Tyto styly pomohly k oživení aplikace. Vzhledem k zaměření této aplikace jsem se rozhodl nevkládat do bakalářské práce části CSS. Vše je však dohledatelné v repositáři na gitHubu. Zároveň jsem tuto aplikaci pomocí stylů upravil tak, aby byla zobrazitelná na co nejvíce zařízeních a tím jsem zařídil její responzivitu. Také jsem do aplikace přidal balíček animate.css, který mi umožnil přidat jednoduché animace na tlačítka, což opět zařídilo větší živost a reaktivnost aplikace.

Odkaz na repositář: https://github.com/martinnovak22/speech_recognition

5.5 Hlavní funkce aplikace

V této části bakalářské práce popíšu hlavní funkcionality mé aplikace. Veškeré dění aplikace jsem vytvořil za pomoci JavaScriptu. V projektu se nachází velká škála funkcí, některé jsou pro technologii SpeechRecognition důležité, jiné jsou pouze pomocné pro lepší práci a úhlednější kód. V této práci bude k nalezení to důležité k rozhraní, které mé aplikaci vládne, vše ostatní bude opět dohledatelné v mém repositáři na gitHubu.

5.5.1 SpeechRecognition v kódu

Jako první jsem se rozhodl uchopit samotné rozhraní SpeechRecognition, protože je v obou demonstrativních komponentách použito podobně. Následně rozeberu každou komponentu a její JavaScript zvlášť.

Použití rozhraní SpeechRecognition není žádná raketová věda, kompletní deklarace také záleží na potřebách projektu. V mém projektu jsem se rozhodl vypisovat průběžně slyšená slova, která rozhraní rozklíčovalo, protože to zlepšuje práci s rozhraním a dává to uživateli pocit, že se něco děje a aplikace není mrtvá. U rozcestníku jsem ponechal defaultní jazyk uživatele. U diktafonu je možnost si jazyk nastavit.

```
1 window.SpeechRecognition =
2   window.SpeechRecognition ||
      window.webkitSpeechRecognition;
3
4 const recognition = new SpeechRecognition();
5 recognition.interimResults = true;
```

Příklad 17: Kompletní deklarace SpeechRecognition

Po deklaraci rozhraní už se představivosti meze nekladou. Pro svůj projekt jsem se rozhodl hlavní funkce navázat na tlačítka, tudíž jako první jsem přidal tlačítko, které spouští odposlech. Na tlačítko jsem přidal eventListener, který čeká na klik. Po kliku se spouští odposlech, tlačítko se deaktivuje za pomoci atributu disabled, a to proto, abych zamezil násobnému klikání na tlačítko, kvůli kterému by mohlo dojít k neočekávaným situacím či errorům, které by mohly uživatele vyvést z míry. Také zde pracuji s konkrétními classami, které zastupují animaci pro dané tlačítko. Spouštěcí tlačítka fungují v obou komponentách podobně, jen pracují s jinými classami. Hlavním rozdílem je to, že spouštěcí tlačítko diktafonu má svůj protějšek, tudíž tlačítko zastavující odposlech. U rozcestníku se odposlech přeruší po prvním odchyceném slově, větě, či frázi.

```
start_button.addEventListener("click", () => {
  recognition.start();
  start_button.disabled = true;
  start_button.classList.add(
    "animate__animated","animate__pulse",
    "animate__infinite","infinite"
  );
});
```

Příklad 18: Ukázka spouštění odposlechu

Spuštění odposlechu je důležité, avšak následně je třeba tento odposlech nějak využít. SpeechRecognition má svoje typy eventListenerů, a tím hlavním je event "result". Tohoto eventu jsem využil v obou demonstrativních komponentách, počáteční deklarace je stejná, následuje však jiná funkcionalita, kterou následně rozeberu podrobně, pro každou komponentu zvlášť.

```
recognition.addEventListener("result", (e) => {
// functionality
});
```

Příklad 19: Deklarace výsledku odposlechu

Druhým důležitým eventem, který jsem využil pouze pro diktafon, je event "end". Tohoto eventu jsem využil proto, abych mohl ovládat kdy chci pokračovat v zadávání, a kdy ho chci ukončit. Toho dostahuji za použití vyhledávacího parametru, který ovládám z potřebných tlačítek. Zároveň jsem do tohoto eventu umístil promazávání pole, ve kterém se ukazují průběžné výpisy odposlechu. V průběhu odposlechu se mění innerHTML, po stisku ukončovacího tlačítka je však třeba pole ručně přemazat. Okamžité vyčištění pole mi přišlo dost násilné, proto jsem se rozhodl přidat timeout, který dává uživateli větší plynulost při práci s aplikací.

```
recognition.addEventListener("end", () => {
  if (searchParams.get("end") === "true") {
    recognition.stop();
    setTimeout(() => {
      texts.innerHTML = "";
    }, 1000);
    return;
  }
  recognition.start();
});
```

Příklad 20: Deklarace ukončení odposlechu

5.5.2 Funkce rozcestníku

Jak už z názvu vyplývá, tuto komponentu jsem vytvořil proto, aby mi umožnila dostat se pomocí hlasu někam dál. Tudíž hlavní funkcí rozcestníku je funkce "open". Tato funkce pracuje s daty uloženými v localStorage prohlížeče. V těchto datech jsou k nalezení názvy a odkazy na stránky, které rozcestník umí otevřít. Jak už jsem zmínil dříve, pro práci s těmito daty jsem vytvořil malé rozhraní, které umožňuje přidat či odebrat stránku a samozřejmě vypsat list s těmi aktuálními.

```
if (text.includes("open")) {
    const storage_list =
        window.localStorage.getItem("sites");
    JSON.parse(storage_list).map((site) => {
        if (text.includes(site.name)) {
            window.open(site.url);
        }
    });
}
```

Příklad 21: Funkce open

Pro lepší demonstraci a zpestření rozcestníku jsem se rozhodl naplnit ho více funkcemi. Některé z těchto funkcí využívají další externí rozhraní. Pro jednu ze schopností rozcestníku jsem využil API na počasí, které mi navrátí aktuální hodnoty, pro místo, kde se uživatel nachází, a to za použití dalšího API, kterým je Geolocation API. K tomuto dojde na základě hlasového příkazu "what's the weather". Další funkcí využívající geolokaci je hlasový příkaz "show me my location". Tímto příkazem se otevře layout s vloženou mapou, která bude ukazovat aktuální pozici uživatele.

Vzhledem k tomu, že jsem fanouškem kryptoměn, rozhodl jsem se přidat i hlasový příkaz "bictoin price now", který zavolá na rozhraní firmy Binance, jedné z největších světových krypto burz, a ta vrátí aktuální tržní cenu bitcoinu.

```
1  if (text.includes("bitcoin price now")) {
2      spinner.removeAttribute("hidden");
3      fetch("https://api.binance.com/api/v3/avgPrice?
           symbol=BTCUSDT")
4       .then((response) => {
5         return response.json();
6       })
7       .then((data) => {
8         console.log(data.price);
9         const btcPriceSpan = document.createElement("span");
10        btcPriceSpan.innerText =
11          "BTC: " + Number(data.price).toFixed(2) + " $";
12        spinner.setAttribute("hidden", "");
13        output.appendChild(btcPriceSpan);
14      });
15 }
```

Příklad 22: Funkce bitcoin price now

Pro lepší pocit z plynulosti aplikace využívám spinner, který se zobrazí, pokud dochází k fetchování dat. Následně už jen vyčkám na příchozí data a

vytvořím z nich JSON, abych s nimi mohl dále lépe pracovat. Ve zbytku funkce už pouze vykresluji výsledek do elementu a opět schovávám spinner. Všechny funkce, které si tahají data z externích rozhraní a vypisují se do elementu pracují na podobné bázi jako lze vidět na předešlé ukázce kódu.

Pro větší všestrannost jsem přidal i funkci search. Tato funkce pouze vezme uživatelem vyřknutá slova a předá je jako query parametr do odkazu pro vyhledávání v googlu. Jednoduchý, avšak mocný hlasový příkaz.

```
1 if (text.includes("search")) {
2     window.open(`https://www.google.com/search?q=${
          text.slice(6)}`);
3 }
```

Příklad 23: Funkce search

5.5.3 Funkce diktafonu

Nejlépe lze však poznat moc SpeechRecognition API na druhé demonstrativní komponentě, kterou je diktafon. Tento diktafon totiž nabízí přepis hlasu z velké škály světových jazyků. Zpracování textu je vždy na rozhraní, mým úkolem bylo v tomto případě brát text a umisťovat ho do pole pro další práci. Text jsem vždy navazoval na sebe, a na konec odchycené fráze jsem vždy přidal tečku a mezeru, pro lepší čitelnost finálního textu. Zároveň jsem si při práci s textem všiml, že ne vždy rozhraní určí, zdali má být první písmeno velké. Proto jsem si vytvořil jednoduchou pomocnou funkci, která mi vždy první písmeno zvětší.

```
1 recognition.addEventListener("result", (e) => {
2   texts.appendChild(p);
3   const text = Array.from(e.results)
4     .map((result) => result[0])
5     .map((result) => result.transcript)
6     .join("");
```

```
7
8     const upperText = firstLetterUpper(text);
9     p.innerText = upperText;
10    if (e.results[0].isFinal) {
11      textArea.appendChild(document.createTextNode(upperText
            + ". "));
12    }
13  });
```

Příklad 24: Práce s textem v diktafonu

Důležitým úkolem bylo také připravit menu, ve kterém si uživatel bude schopen jazyk změnit. Toho jsem dosáhl pomocí jednoduchého selektivního menu, které při změně vezme hodnotu aktuálně zvoleného pole a pomocí toho jazyk přenastaví.

```
1  language_option.addEventListener("change", (e) => {
2    recognition.lang = e.target.value;
3  });
```

Příklad 25: Změna jazyka pomocí selektivního menu

Rozhraní Speech má opravdu velkou škálu jazyků. Pro většinu jazyků dokonce existují i krajinné dialekty či vlastní státní verze jazyků, jako je například španělština ve Španělsku nebo v Brazílii. Já jsem se pro svůj projekt rozhodl vyselektovat jen určitě množství, u kterého jsem si byl jistý, že budu schopen dohledat i hlas, který bude schopen text zpětně přečíst. Zároveň jsem zachoval vždy jen jednu verzi jazyka. Pro jazyky jsem si připravil soubor, ve kterém jsem vytvořil velké pole, které obsahuje vždy název jazyka a jeho zkratku, pomocí které se následně nastavuje jazyk diktafonu.

```
export const languages = [
  ["Afrikaans", "af-ZA"],
  ["Arabic", "ar-001"],
  ["Bulgarian", "bg-BG"],
  ["Chinese (Simp.)", "cmn-Hans-CN"],
  ["Croatian", "hr_HR"],
  ["Czech", "cs-CZ"],
  ["English", "en-GB"],
  ["French", "fr-FR"],
  ["German", "de-DE"],
  ["Greek", "el-GR"],
  ["Finnish", "fi-FI"],
  ["Hebrew", "he-IL"],
  ["Hindi", "hi-IN"],
  ["Hungarian", "hu-HU"],
  ["Indonesian", "id-ID"],
  ["Italian", "it-IT"],
  ["Japanese", "ja-JP"],
  ["Korean", "ko-KR"],
  ["Dutch", "nl-NL"],
  ["Norwegian", "nb-NO"],
  ["Polish", "pl-PL"],
  ["Portuguese", "pt-PT"],
  ["Romanian", "ro-RO"],
  ["Russian", "ru-RU"],
  ["Slovak", "sk-SK"],
  ["Spanish", "es-ES"],
  ["Swedish", "sv-SE"],
  ["Thai", "th-TH"],
  ["Turkish", "tr-TR"],
  ["Ukrainian", "uk-UA"],
  ["Vietnamese", "vi-VN"],
  ["Zulu", "zu-ZA"],
];
```

Příklad 26: Jazyky použité v diktafonu

Jako další jsem přidal funkci kopírování. Tu jsem upravil tak, aby mohla správně fungovat, jak na počítači, tak na mobilu. Mobil totiž vyžaduje přesnou selekci textu, aby ho mohl schovat do schránky, toto lze vidět na následující ukázce kódu pod komentářem "For mobile".

```
copy_button.addEventListener("click", () => {
  // For mobile
  textArea.select();
  textArea.setSelectionRange(0, 99999);

  navigator.clipboard.writeText(textArea.value).then(() =>
     {
    addBounce(copy_button);
  });
});
```

Příklad 27: Kopírování textu

Jak už jsem v této práci psal, součástí rozhraní Speech je i takzvaná syntéza slova. SpeechSynthesis umožňuje vzít text a přetvořit ho na hlas. Každý jazyk však vyžaduje svůj hlas, a to hlavně kvůli výslovnosti jazykových specifik, jako jsou například přehlásky v němčině.

```
read_button.addEventListener("click", () => {
  searchParams.set("end", "true");
  recognition.stop();
  start_Button.disabled = false;
  addBounce(read_button);
  const msg = textArea.value;
  const speech = new SpeechSynthesisUtterance(msg);
  speech.voice = getVoicebyLang(language_option.value.slice
     (-2));
  synth.speak(speech);
});
```

Příklad 28: SpeechSynthesis

5.6 Pomocné funkce aplikace

Pro přehlednost svého kódu jsem si vytvořil spoustu pomocných funkcí. Jednou ze zmíněných je funkce, která mi zvětší první písmeno předaného textu.

```
export const firstLetterUpper = (text) => {
  return text.charAt(0).toUpperCase() + text.slice(1);
};
```

Příklad 29: Funkce pro zvětšení prvního písmena

Další velmi důležitou pomocnou funkcí je funkce, která vrátí aktuální pozici uživatele pomocí geolocation API, zabudované v prohlížeči. Zároveň dojde k výpisu do konzole. Toto jsem se v kódu rozhodl ponechat, protože mi to umožňuje lepší demonstraci celého průběhu funkce.

```
export const getLocation = () =>
  new Promise((resolve) => {
    function success(pos) {
      const crd = pos.coords;
      console.log("Your current position is:");
      console.log(`Latitude : ${crd.latitude}`);
      console.log(`Longitude: ${crd.longitude}`);
      console.log(`More or less ${crd.accuracy} meters.`);

      resolve(crd);
    }

    function error(err) {
      console.warn(`ERROR(${err.code}): ${err.message}`);
    }

    navigator.geolocation.getCurrentPosition(success, error
        , OPTIONS);
  });
```

Příklad 30: Funkce vracející lokaci uživatele

Jako poslední bych rád zmínil funkci, která je vidět v mnoha ukázkách kódu. Pomocí této funkce dodávám aplikaci život. Tato funkce pouze přidá animační classu a následně ji po chvíli zase odebere. Tím dojde k zakymácení tlačítka a uživateli to dodá pocit, že na něj aplikace reaguje a něco se děje.

```
1 export const addBounce = (element) => {
2   element.classList.add("animate__animated", "
        animate__bounceIn");
3
4   setTimeout(() => {
5     element.classList.remove("animate__animated", "
          animate__bounceIn");
6   }, 1000);
7 };
```

Příklad 31: Animační funkce

6 Závěr

Bakalářská práce se zabývá technologií Speech Recognition. Tato technologie slouží k rozpoznávání slov z hlasu člověka.

V teoretické části jsem se zaměřil na představení samotné technologie, kde se využívá a jaké jsou její klíčové vlastnosti. Také jsem vytyčil všechny pojmy pro práci s touto technologií, a to ať už se jednalo o kostru aplikace, či o rozhraní, která rozšiřují schopnosti aplikace. V poslední řadě jsem se zaměřil na prozkoumání výhod a nevýhod této technologie.

V praktické části bakalářské práce jsem vytvořil dvě demonstrativní komponenty. Jednou z nich je funkční rozcestník, který umožňuje otevírat stránky pomocí slovního příkazu. Toto mi přišlo příliš banální, proto jsem se rozhodl rozcestník obohatit o další slovní příkazy, které tak rozšiřují funkčnost rozcestníku a dělají z něj jednoduchého hlasového asistenta. Druhou komponentou je diktafon, který umožňuje hlasově zadávat text v mnoha světových jazycích. Zároveň umožňuje text okopírovat či zpětně přečíst.

Technologie Speech Recognition nepatří mezi novinky dnešní doby, avšak podpora této technologie vázne a je třeba rozhraní stále cvičit a zlepšovat jeho rozpoznávání slov. I tak si myslím, že tato technologie má velkou sílu, kterou v dnešní době využívají hlavně profesionálně vyvinuté aplikace, které fungují jako hlasoví asistenti.

Závěrem bych rád řekl, že moje zkušenost s touto technologií byla velice kladná. Velmi mě překvapilo, jako komplexní slova je rozhraní schopné rozklíčovat a jsem celkem v údivu, že se webové rozhraní na rozpoznávání řeči nevyužívá víc. Velkým problémem je zřejmě strach o soukromí a bezpečnost uživatelských dat.

Seznam použité literatury a zdrojů

[1] MDN Web Docs. *HTML: HyperText Markup Language* [online]. 1 [cit. 2023-03-30]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTML

[2] MDN Web Docs. *HTML5* [online]. 1 [cit. 2023-03-30]. Dostupné z: https://developer.mozilla.org/en-US/docs/Glossary/HTML5

[3] MDN Web Docs. *APIs in HTML5* [online]. 1 [cit. 2023-03-30]. Dostupné z: https://www.geeksforgeeks.org/explain-apis-available-in-html5/

[4] MDN Web Docs. *CSS: Cascading Style Sheets* [online] . 1 [cit. 2023-03-30]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/CSS

[5] MDN Web Docs. *CSS: Responsive design* [online]. 1 [cit. 2023-03-30]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Responsive_Design

[6] MDN Web Docs *What is JavaScript?* [online]. 1 [cit. 2023-03-30]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript

[7] Geegs for geeks *JavaScript Tutorial* [online]. 1 [cit. 2023-03-30]. Dostupné z: https://www.geeksforgeeks.org/javascript/?ref=lbp

[8] Geegs for geeks *Introduction to JavaScript* [online]. 1 [cit. 2023-03-30]. Dostupné z: https://www.geeksforgeeks.org/introduction-to-javascript/?ref=lbp

[9] IBM *What is API?* [online]. 1 [cit. 2023-03-30]. Dostupné z: https://www.ibm.com/topics/api

[10] IBM *What is Speech Recognition?* [online]. 1 [cit. 2023-03-30]. Dostupné z: https://www.ibm.com/topics/speech-recognition

[11] MDN Web Docs. *Web Speech API* [online]. 1 [cit. 2023-03-30]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API

[12] MDN Web Docs. *Speech Recognition* [online]. 1 [cit. 2023-03-30]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/SpeechRecognition

[13] MDN Web Docs. *Speech Synthesis* [online]. 1 [cit. 2023-03-30]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/SpeechSynthesis

[14] MDN Web Docs. *Web Speech API use* [online]. 1 [cit. 2023-03-30]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API/Using_the_Web_Speech_API

[15] Codecademy. *What is and IDE?* [online]. 1 [cit. 2023-03-30]. Dostupné z: https://www.codecademy.com/article/what-is-an-ide

Seznam příkladů

Seznam obrázků

Příloha

Odkaz na repositář s praktickou částí:

https://github.com/martinnovak22/speech_recognition

Odkaz na samotnou aplikaci:

https://rozcestnik.netlify.app/