// breakSentence.js

import { qs } from "./util";

document.addEventListener('DOMContentLoaded', function() {

const textElem = qs('#textOriginalArea');

const outputArea = qs('#sentenceListArea');

const buttonElem = qs('#breakBtn');

buttonElem.addEventListener('click', function(event) {

event.preventDefault(); // stop the button from submitting

const text = textElem.textContent;

const regex = /[^.!?]+[.!?]+/g; //not start with .!? but end with .!?

const sentences = text.match(regex) || [];

const trimmedsentences = sentences.map((sentence) => sentence.trim());

outputArea.innerHTML = trimmedsentences.map(sentence => "➢" + sentence).join('<br>');

});

})

//englishDefinition.js

import { qs } from "./util";

const url = 'https://twinword-word-graph-dictionary.p.rapidapi.com/definition/?entry=mask';

const options = {

method: 'GET',

headers: {

'X-RapidAPI-Key': '1b58f68877msh09e7f1c9031ff15p14b078jsnbd682f41b858',

'X-RapidAPI-Host': 'twinword-word-graph-dictionary.p.rapidapi.com'

}

};

try {

const response = await fetch(url, options);

const result = await response.text();

console.log(result);

} catch (error) {

console.error(error);

}

// lookUp.js

import { qs } from "./util";

document.addEventListener('DOMContentLoaded', function() {

const buttonElem = qs('#lookUpBtn');

const outputArea = qs('#notesArea');

const checkboxChinese = qs('#chinese-meaning');

const checkboxEnglish = qs('#english-meaning');

const checkboxExample = qs('#example-sentence');

buttonElem.addEventListener('click', async (event) => {

event.preventDefault(); // stop the button from submitting

var selection = document.getSelection().toString().trim();

console.log(selection);

console.log(buttonElem);

if (!selection) {

alert('請先選取字再查單詞');

return;

}

if (!checkboxChinese.checked && !checkboxEnglish.checked && !checkboxExample.checked) {

alert('勾選「中文」、「English」、「例句」');

return;

}

let divElem = document.createElement('div'); // note section

divElem.classList.add('note');

divElem.setAttribute('contenteditable', 'true');

divElem.innerHTML = `<strong>${selection}</strong><br>`;

if (checkboxChinese.checked) {

divElem = await chineseMeaning(selection, divElem);

};

if (checkboxEnglish.checked) {

divElem = await englishMeaning(selection, divElem);

};

outputArea.appendChild(divElem);

});

async function chineseMeaning(word, divElem) {

const url = 'https://microsoft-translator-text.p.rapidapi.com/Dictionary/Lookup?to=zh-Hans&api-version=3.0&from=en';

const options = {

method: 'POST',

headers: {

'content-type': 'application/json',

'X-RapidAPI-Key': '1b58f68877msh09e7f1c9031ff15p14b078jsnbd682f41b858',

'X-RapidAPI-Host': 'microsoft-translator-text.p.rapidapi.com'

},

body: JSON.stringify([{ Text: word }])

};

try {

const response = await fetch(url, options);

if (!response.ok) {

throw new Error(`HTTP error! Status: ${response.status}`);

}

const result = await response.json();

if (result[0].displaySource && result[0].translations.length > 0) {

// const selectedWord = result[0].displaySource;

let translations = result[0].translations;

let meanings = translations.map(translation => translation.displayTarget);

divElem.innerHTML += `

${meanings.join(', ')}

<br>

`;

return divElem;

} else {

alert('請完整選擇一個字詞')

}

} catch (error) {

console.error('Error: ', error.message);

outputArea.textContent = "找不到相關翻譯"

}

}

async function englishMeaning(word, divElem) {

const url = `https://twinword-word-graph-dictionary.p.rapidapi.com/definition/?entry=${word}`;

const options = {

method: 'GET',

headers: {

'X-RapidAPI-Key': '1b58f68877msh09e7f1c9031ff15p14b078jsnbd682f41b858',

'X-RapidAPI-Host': 'twinword-word-graph-dictionary.p.rapidapi.com'

}

};

try {

const response = await fetch(url, options);

if (!response.ok) {

throw new Error(`HTTP error! Status: ${response.status}`);

}

const result = await response.json();

const meanings = result.meaning || {}; // meanings is a key-value object

console.log("result: ", result);

console.log("result-meaning: ", result.meaning);

let formattedMeanings = Object.entries(meanings)

.filter(([key, value]) => value) // filter out empty meanings

.map(([key, value]) => {

value = value.replace(/\n/g, '<br>'); // replace \n with <br>

return value;

})

.join('<br>');

divElem.innerHTML += `

${formattedMeanings}

`;

return divElem;

} catch (error) {

console.error(error);

}

}

});

// main.js

import { qs } from "./util";

document.addEventListener('DOMContentLoaded', function() {

// ----- Clear the textarea -----

const clearOriginalBtn = qs('#clearOriginalBtn');

const textOriginalArea = qs('#textOriginalArea');

clearOriginalBtn.addEventListener('click', () => textOriginalArea.textContent = "");

const clearListBtn = qs('#clearListBtn');

const sentenceListArea = qs('#sentenceListArea');

clearListBtn.addEventListener('click', () => sentenceListArea.textContent = "");

const clearTranslationBtn = qs('#clearTranslationBtn');

const translationArea = qs('#translationArea');

clearTranslationBtn.addEventListener('click', () => translationArea.textContent = "");

const clearNotesBtn = qs('#clearNotesBtn');

const notesArea = qs('#notesArea');

clearNotesBtn.addEventListener('click', () => notesArea.textContent = "");

// ----- End of clear the textarea -----

// ----- Toggle the hint -----

const originalHint = qs('#originalHint');

const originalQ = qs('#originalQ');

const listHint = qs('#listHint');

const listQ = qs('#listQ');

const translationHint = qs('#translationHint');

const translationQ = qs('#translationQ');

const notesHint = qs('#notesHint');

const notesQ = qs('#notesQ');

originalQ.addEventListener('click', () => {

if (originalHint.style.display === 'block') {

originalHint.style.display = 'none';

} else {

originalHint.style.display = 'block';

}

});

listQ.addEventListener('click', () => {

if (listHint.style.display === 'block') {

listHint.style.display = 'none';

} else {

listHint.style.display = 'block';

}

});

translationQ.addEventListener('click', () => {

if (translationHint.style.display === 'block') {

translationHint.style.display = 'none';

} else {

translationHint.style.display = 'block';

}

});

notesQ.addEventListener('click', () => {

if (notesHint.style.display === 'block') {

notesHint.style.display = 'none';

} else {

notesHint.style.display = 'block';

}

});

// ----- End of show the hint -----

});

// savePDF.js

import { qs } from "./util";

import { jsPDF } from "jspdf";

const saveBtnElem = qs('#saveBtn');

function saveAsPDF() {

// ----- basic pdf setting -----

const pdf = new jsPDF({

lineHeight: 1

});

pdf.setFont('NotoSansTC-VariableFont\_wght'); // set Chinese font

const pageWidth = pdf.internal.pageSize.width;

const pageHeight = pdf.internal.pageSize.height;

const margins = {left: 10, right: 10, top: 10, bottom: 10};

const maxWidth = pageWidth - (margins.left + margins.right);

// ----- end of basic pdf setting -----

// ----- generate translation section -----

const translationContent = qs('#translationArea').innerText;

const splitTranslation = pdf.splitTextToSize(translationContent, maxWidth); // split text to fit within the maxWidth

console.log("Translations: ", splitTranslation);

let currentY = margins.top; // starting Y for translation header

pdf.text('Translations:', margins.left, currentY); // add header

currentY += 10; // starting Y for translation content

for (let line of splitTranslation) {

if (currentY + 10 > pageHeight - margins.bottom) { // ensure enough space for new content

pdf.addPage();

currentY = margins.top; // reset margin top for new page

}

pdf.text(line, margins.left, currentY); // output translation content

currentY += 10 // increase by line height

}

// ----- end of generate translation section -----

currentY += 5; // gap between sections

// ----- generate notes section -----

if (currentY + 10 > pageHeight - margins.bottom) { // ensure enough space for new content

pdf.addPage();

currentY = margins.top; // reset margin top for new page

}

pdf.text('Note: ', margins.left, currentY);

currentY += 10; // starting Y for notes content

const notesContent = qs('#notesArea').innerText;

const splitNotes = pdf.splitTextToSize(notesContent, maxWidth); // split text to fit within the maxWidth

console.log("notes: ", splitNotes)

for (let line of splitNotes) {

if (currentY +10 > pageHeight - margins.bottom) {

pdf.addPage();

currentY = margins.top;

}

pdf.text(line, margins.left, currentY);

currentY += 10;

}

// ----- end of generate notes section -----

pdf.save('句句通.pdf');

}

saveBtnElem.addEventListener('click', saveAsPDF)

// translate.js

import { qs } from "./util";

document.addEventListener('DOMContentLoaded', function() {

const textElem = qs('#sentenceListArea');

const buttonElem = qs('#translateBtn');

const outputArea = qs('#translationArea');

buttonElem.addEventListener('click', async (event) => {

event.preventDefault(); // stop the button from submitting

const htmlContent = textElem.innerHTML;

const parts = htmlContent.split('<br>');

const sentences = [];

parts.forEach(part => {

const trimmedPart = part.trim();

if (trimmedPart) {

sentences.push(part);

}

});

outputArea.innerHTML = ""; // clear output area before retrieving new data

loading();

let resultHtml = '';

for (let sentence of sentences) {

resultHtml += await translate(sentence);

}

outputArea.innerHTML = resultHtml;

});

async function translate(sentence) {

// ----- api information -----

const url = 'https://microsoft-translator-text.p.rapidapi.com/translate?to%5B0%5D=zh-Hant&api-version=3.0&profanityAction=NoAction&textType=plain';

const options = {

method: 'POST',

headers: {

'content-type': 'application/json',

'X-RapidAPI-Key': '1b58f68877msh09e7f1c9031ff15p14b078jsnbd682f41b858',

'X-RapidAPI-Host': 'microsoft-translator-text.p.rapidapi.com'

},

body: JSON.stringify([

{

Text: sentence

}

])

};

// ----- end of api information -----

try {

const response = await fetch(url, options);

if (!response.ok) {

throw new Error(`HTTP error! Status: ${response.status}`);

}

const result = await response.json();

return `<span class="original-text">${sentence}</span><br>` +

`<span class="translated-text">${result[0].translations[0].text}</span><hr>`;

} catch (error) {

console.error("Translation error: ", error);

return `<span class="translate-error">Error translating sentence: ${sentence}</span><br><hr>`;

}

}

function loading() {

outputArea.textContent = `Generating translation. Please wait.`;

return outputArea;

}

});

// util.js

export function qs(selector) {

return document.querySelector(selector);

}

// index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>句句通</title>

<link rel="stylesheet" href="/style.css">

</head>

<body>

<header>

把英文文章分句翻譯，也可以查詢單詞意思，最後把翻譯和查詢的資料編輯後變成pdf檔儲存。<br>

請按照以下的步驟1→2→3→4完成你的筆記。

</header>

<section class="card">

<h1>1. 輸入文字

<svg id="originalQ" xmlns="http://www.w3.org/2000/svg" width="1em" height="1em" viewBox="0 0 256 256">

<path fill="currentColor" d="M140 180a12 12 0 1 1-12-12a12 12 0 0 1 12 12M128 72c-22.06 0-40 16.15-40 36v4a8 8 0 0 0 16 0v-4c0-11 10.77-20 24-20s24 9 24 20s-10.77 20-24 20a8 8 0 0 0-8 8v8a8 8 0 0 0 16 0v-.72c18.24-3.35 32-17.9 32-35.28c0-19.85-17.94-36-40-36m104 56A104 104 0 1 1 128 24a104.11 104.11 0 0 1 104 104m-16 0a88 88 0 1 0-88 88a88.1 88.1 0 0 0 88-88" />

</svg>

</h1>

<div id="originalHint" class="hintBox">

<ul>

<li>原文限英文。</li>

<li>單句長度不可超過10,000字母（含空格）</li>

<li>總長度不可超過100句，也不可超過50,000字母（含空格）</li>

</ul>

</div>

<div contenteditable="true" class="area" id="textOriginalArea">請輸入英文原文</div>

<div>

<button type="button" id="clearOriginalBtn">清除原文</button>

<button type="button" id="breakBtn">陳列句子</button>

</div>

</section>

<section class="card">

<h1>2. 陳列句子

<svg id="listQ" xmlns="http://www.w3.org/2000/svg" width="1em" height="1em" viewBox="0 0 256 256">

<path fill="currentColor" d="M140 180a12 12 0 1 1-12-12a12 12 0 0 1 12 12M128 72c-22.06 0-40 16.15-40 36v4a8 8 0 0 0 16 0v-4c0-11 10.77-20 24-20s24 9 24 20s-10.77 20-24 20a8 8 0 0 0-8 8v8a8 8 0 0 0 16 0v-.72c18.24-3.35 32-17.9 32-35.28c0-19.85-17.94-36-40-36m104 56A104 104 0 1 1 128 24a104.11 104.11 0 0 1 104 104m-16 0a88 88 0 1 0-88 88a88.1 88.1 0 0 0 88-88" />

</svg>

</h1>

<div id="listHint" class="hintBox">

<ul>

<li>一次最多翻譯25句，請把不需要翻譯的句子刪除。</li>

<li>總長度不可超過5,000字母（含空格）。</li>

</ul>

</div>

<div contenteditable="true" class="area" id="sentenceListArea">

➢範例句。<br>

➢範例句。<br>

➢範例句。<br>

</div>

<div>

<button type="button" id="clearListBtn">清除全部句子</button>

<button type="button" id="translateBtn">翻譯</button>

</div>

</section>

<section class="card">

<h1>3. 翻譯

<svg id="translationQ" xmlns="http://www.w3.org/2000/svg" width="1em" height="1em" viewBox="0 0 256 256">

<path fill="currentColor" d="M140 180a12 12 0 1 1-12-12a12 12 0 0 1 12 12M128 72c-22.06 0-40 16.15-40 36v4a8 8 0 0 0 16 0v-4c0-11 10.77-20 24-20s24 9 24 20s-10.77 20-24 20a8 8 0 0 0-8 8v8a8 8 0 0 0 16 0v-.72c18.24-3.35 32-17.9 32-35.28c0-19.85-17.94-36-40-36m104 56A104 104 0 1 1 128 24a104.11 104.11 0 0 1 104 104m-16 0a88 88 0 1 0-88 88a88.1 88.1 0 0 0 88-88" />

</svg>

</h1>

<div id="translationHint" class="hintBox">

<ul>

<li>選取要查詢的單字或片語，按「查單詞」。</li>

<li>查詢的相關資料會出現在下方筆記欄。</li>

</ul>

</div>

<div contenteditable="true" class="area" id="translationArea">

<span class="original-text">➢原句。</span><br>

<span class="translated-text">➢翻譯。<br></span>

<span class="original-text">➢原句。</span><br>

<span class="translated-text">➢翻譯。<br></span>

<span class="original-text">➢原句。</span><br>

<span class="translated-text">➢翻譯。<br></span>

</div>

<div>

<button type="button" id="clearTranslationBtn">清除</button>

<button type="button" id="lookUpBtn">查單詞</button>

<input type="checkbox" id="chinese-meaning"><label for="chinese-meaning" class="lookup-checkbox" >中文</label>

<input type="checkbox" id="english-meaning"><label for="english-meaning" class="lookup-checkbox" >English</label>

<input type="checkbox" id="example-sentence"><label for="example-sentence" class="lookup-checkbox" >例句</label>

</div>

</section>

<section class="card">

<h1>4. 筆記

<svg id="notesQ" xmlns="http://www.w3.org/2000/svg" width="1em" height="1em" viewBox="0 0 256 256">

<path fill="currentColor" d="M140 180a12 12 0 1 1-12-12a12 12 0 0 1 12 12M128 72c-22.06 0-40 16.15-40 36v4a8 8 0 0 0 16 0v-4c0-11 10.77-20 24-20s24 9 24 20s-10.77 20-24 20a8 8 0 0 0-8 8v8a8 8 0 0 0 16 0v-.72c18.24-3.35 32-17.9 32-35.28c0-19.85-17.94-36-40-36m104 56A104 104 0 1 1 128 24a104.11 104.11 0 0 1 104 104m-16 0a88 88 0 1 0-88 88a88.1 88.1 0 0 0 88-88" />

</svg>

</h1>

<div id="notesHint" class="hintBox">

<ul>

<li>可手動增減筆記內容</li>

<li>句末shift + Enter：在單格筆記欄增加一行。</li>

<li>文末Enter：新增一格筆記欄。</li>

<li>del鍵：刪除文字或筆記欄。</li>

<li>筆記生成：把翻譯區域和筆記區域儲存成PDF檔。</li>

</ul>

</div>

<div contenteditable="true" class="area" id="notesArea">

</div>

<div>

<button type="button" id="clearNotesBtn">clear</button>

<button type="button" id="saveBtn">筆記生成</button>

</div>

</section>

<script src="/js/main.js" type="module"></script>

<script src="/js/translate.js" type="module"></script>

<script src="/js/breakSentence.js" type="module"></script>

<script src="/js/lookUp.js" type="module"></script>

<script src="/js/NotoSansTC-VariableFont\_wght-normal.js" type="module"></script>

<script src="/js/savePdf.js" type="module"></script>

</body>

</html>

// openai-test.js

import OpenAI from "openai";

const text = 'Frozen produce is mostly picked and frozen at the peak of ripeness, whereas some fresh produce is picked under ripe. '

const word = 'whereas'

const instructions = "For the word provided from the text, analyze and present the following details: the word itself followed by a colon, the part of speech of the word, abbreviated and enclosed in parentheses, the Traditional Chinese translation, the definition of the word, an original example sentence using the word, which is not derived from the text. Ensure each piece of information is listed on a separate line."

const openai = new OpenAI();

async function main() {

const completion = await openai.chat.completions.create({

messages: [{"role": "system", "content": "You are a helpful assistant."},

{"role": "user", "content": text},

{"role": "assistant", "content": instructions},

{"role": "user", "content": `text: ${text}, word: ${word}`}],

model: "gpt-3.5-turbo",

temperature: 1,

});

console.log(completion.choices[0]);

}

main();

// style.css

@import url('https://fonts.googleapis.com/css2?family=Noto+Sans+SC:wght@100..900&family=Noto+Sans+TC:wght@100..900&family=Noto+Sans:ital,wght@0,100..900;1,100..900&display=swap');

:root {

--milky: #F5F7FA;

--light-blue: #4EA5FF;

--dark-blue: #207BFF;

--tc-font: "Noto Sans TC", sans-serif;

--sc-font: "Noto Sans SC", sans-serif;

--en-font: "Noto Sans", sans-serif;

--space-big: 15px;

--space-small: 5px;

}

\* {

box-sizing: border-box;

}

.noto-sans {

font-family: "Noto Sans", sans-serif;

font-optical-sizing: auto;

font-weight: 400;

font-style: normal;

font-variation-settings:

"wdth" 100;

}

.noto-sans-sc {

font-family: "Noto Sans SC", sans-serif;

font-optical-sizing: auto;

font-weight: 400;

font-style: normal;

}

.noto-sans-tc {

font-family: "Noto Sans TC", sans-serif;

font-optical-sizing: auto;

font-weight: 400;

font-style: normal;

}

body {

display: flex;

flex-direction: column;

justify-content: center;

align-items: center;

font-size: 20px;

font-family: var(--tc-font), var(--sc-font), var(--en-font);

}

header {

width: 50vw;

color: var(--dark-blue);

padding: var(--space-big)

}

.card {

width: 100vw;

display: flex;

flex-direction: column;

align-items: center;

margin: var(--space-big) auto;

}

h1 {

margin: 0;

}

.area {

border: 1px solid #ccc;

width: 80vw;

padding: var(--space-big);

}

.original-text {

color: black;

}

.translated-text {

color: var(--dark-blue);

}

.translate-error {

color: red;

}

#notesArea {

display: grid;

grid-template-columns: repeat(auto-fill, 250px);

grid-gap: var(--space-big);

justify-content: start;

}

#notesArea div, .note {

margin: 0;

border: 1px solid #ccc;

box-shadow: 1px 1px 3px rgba(0, 0, 0, 0.3);

padding: var(--space-small);

font-size: 15px;

}

.hintBox {

display: none;

width: 80vw;

background-color: var(--milky);

font-size: 14px;

}

.lookup-checkbox {

font-size: 14px;

}