

Github URL

Vercel URL

完成的功能

- 清除、複製按鈕
- 翻譯語言下拉選單
- 點擊或拖曳上傳文檔
- 顯示上傳的文檔
- 可以批量翻譯 txt 文檔

未完成的功能

- 因需要 npm docx 檔的翻譯功能未實現

P1:API 來源與功能

- API 網址
<https://rapidapi.com/sibaridev/api/rapid-translate-multi-traduction>
- API 功能說明：
免費的額度是一天 50 次
用 POST 發送 json 給伺服器後會回傳翻譯結果
 - body:
 - from : 輸入文字的語言
 - to : 翻譯結果的語言
 - q : 翻譯文本

```
const url = 'https://rapid-translate-multi-traduction.p.rapidapi.com/t';
const options = {
  method: 'POST',
  headers: {
    'content-type': 'application/json',
    'X-RapidAPI-Key': '563ff64563msh068d946124e802ap1566dcjsn01e552b57193',
    'X-RapidAPI-Host': 'rapid-translate-multi-traduction.p.rapidapi.com'
  },
  body: {
    from: 'en',
    to: 'ar',
    q: 'Hello ! Rapid Translate Multi Traduction'
  }
};
```

P2:實作過程

測試 API

- rapidapi 有提供調用 api 的 code

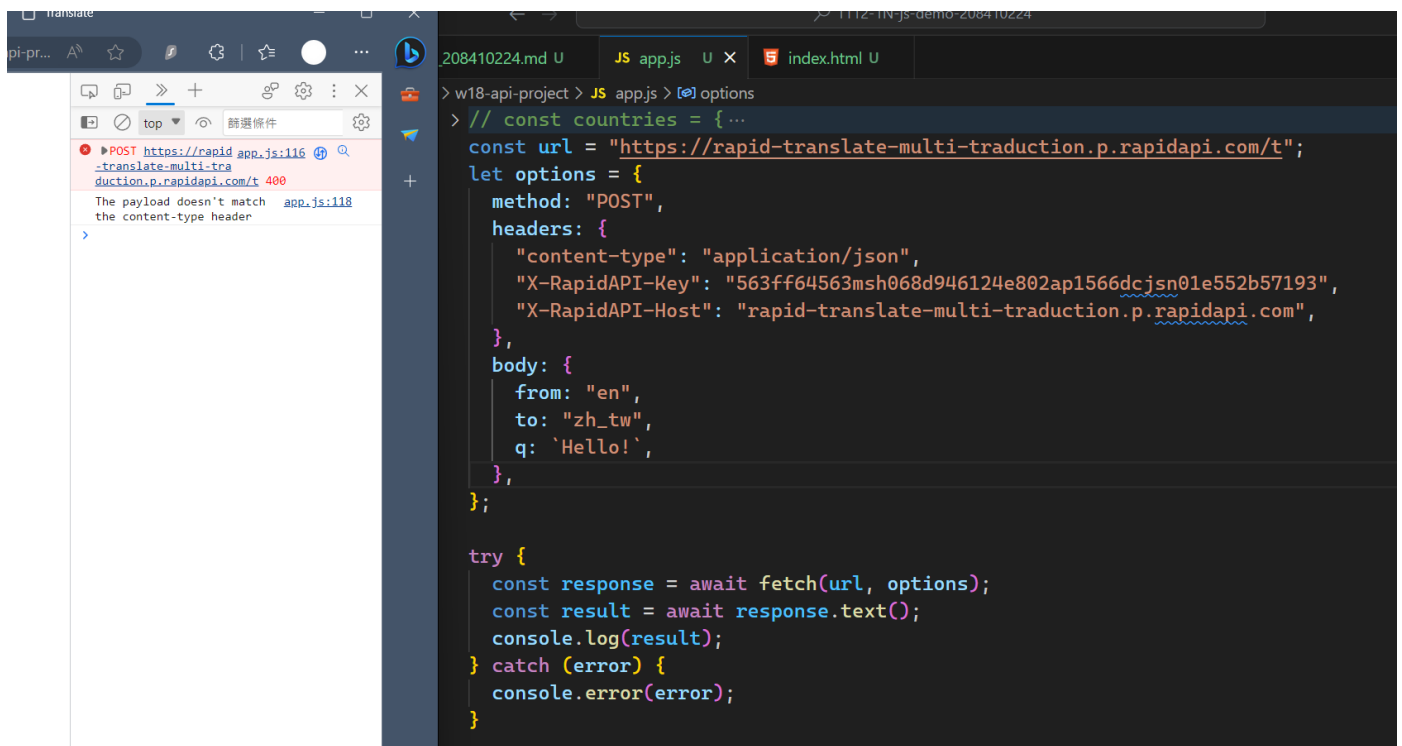
The screenshot displays the RapidAPI interface for the 'Rapid Translate Multi Traduction' API. The 'Test Endpoint' button is highlighted. The 'Code Snippets' tab shows the following JavaScript code:

```
(JavaScript) fetch
Copy Code

const url = 'https://rapid-translate-multi-traduction.p.rapidapi.com/t';
const options = {
  method: 'POST',
  headers: {
    'content-type': 'application/json',
    'X-RapidAPI-Key': '563ff64563msh068d946124e802ap1566dcjsn01e552b57193',
    'X-RapidAPI-Host': 'rapid-translate-multi-traduction.p.rapidapi.com'
  },
  body: {
    from: 'en',
    to: 'ar',
    q: 'Hello ! Rapid Translate Multi Traduction'
  }
};

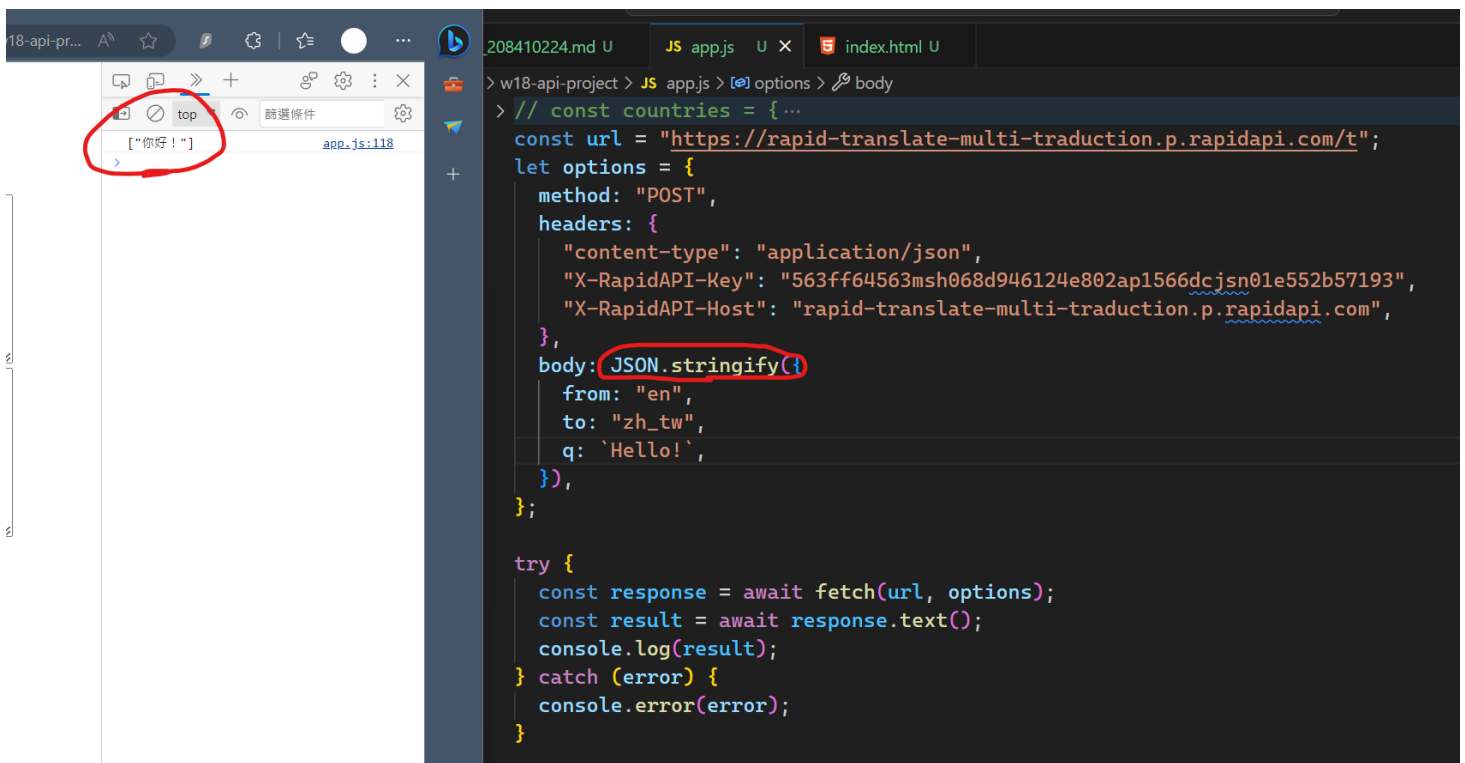
try {
  const response = await fetch(url, options);
  const result = await response.text();
  console.log(result);
} catch (error) {
  console.error(error);
}
```

- 但直接使用並沒有成功，console 會報錯




- 花了一段時間才找到解決方法，需要先將 body 內容轉換成 json 格式伺服器才能讀取

```
body: JSON.stringify({
  from: "en",
  to: "zh_tw",
  q: `Hello!`,
}),
```



知道 api 如何運作後，參考了一些配色網站，寫了 html 和 css



LOLCOLORS

Curated color palette inspiration.

[WebDesignRankings Home](#)

[Best Web Design Companies](#)

[Best SEO Companies](#)

[Agencies By Location](#)

[Agencies By Industry](#)

[Blog](#)


In need of a total site redesign, ground-up build, or other extensive design work? Check out our various lists of the [top web design companies](#) for an objective look at the best web design agencies around.

Looking for help with marketing as well? We've got you covered with our rankings of the [best digital marketing companies](#) in the country!


Sorting and "liking" color palettes is temporarily disabled while we're expanding this resource.

Other free tools:


- [Image to base64 converter](#)
- [CSS Layout Generator](#)
- [Px to percentage calculator](#)
- [Px to em calculator](#)




♥ 4714




♥ 2209




♥ 2877



♥ 2046



♥ 1477



♥ 1369

JSAPI Project 翻譯

輸入要翻譯的文字

翻譯結果

Auto

→

Amharic

翻譯

Copy

Clear

- 到 Font Awesome 找 icon 代替一些文字

API Project

 翻譯

輸入要翻譯的文字

翻譯結果

Auto

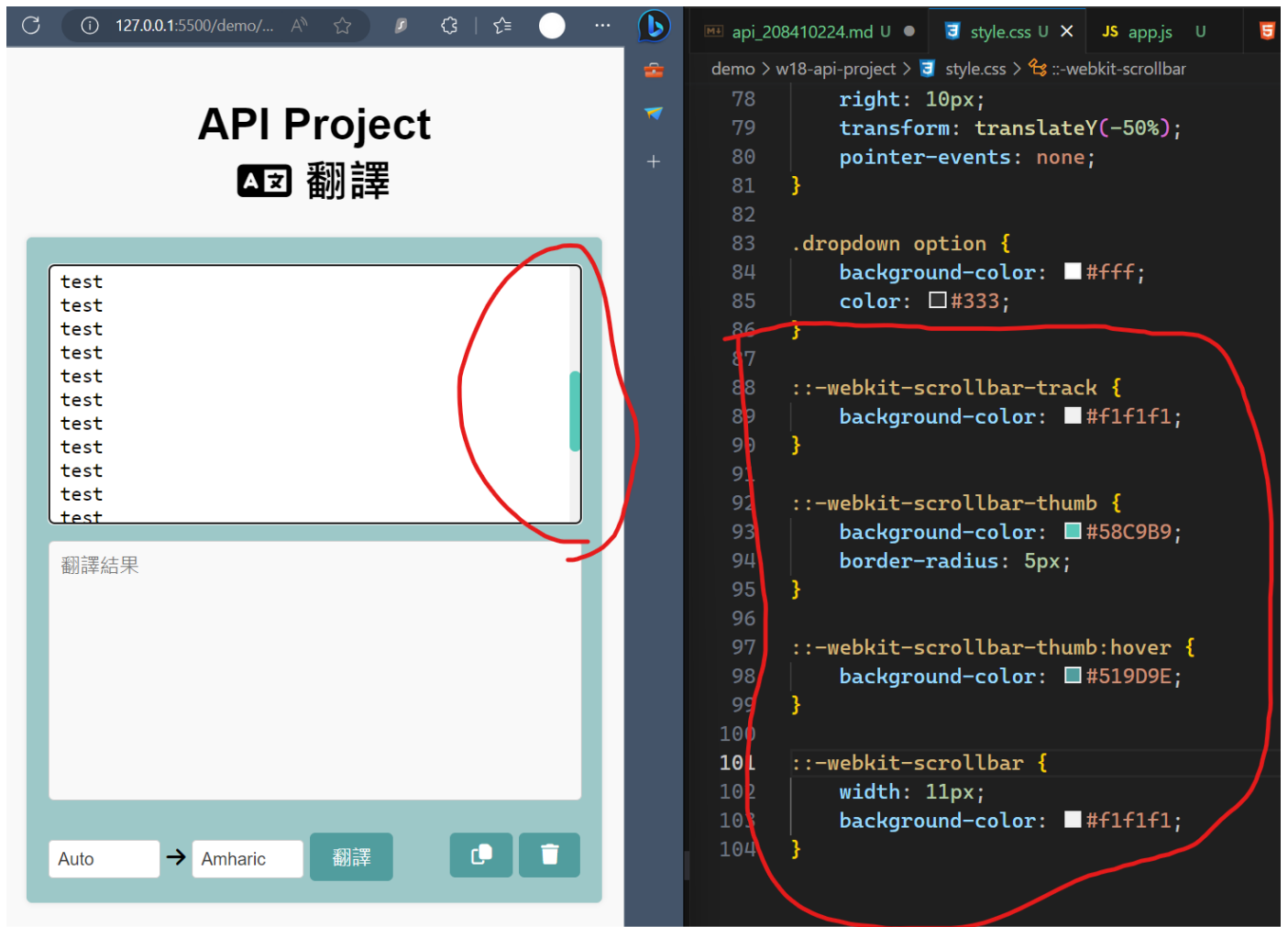


Amharic

翻譯



- 覺得預設的 scrollbar 很醜，改了樣式



UI 寫完後，輪到 JS

- 先把按鈕功能完成

```
114  
115 //transBox  
116 const input = document.querySelector("#input");  
117 const output = document.querySelector("#output");  
118 //btns  
119 const translateBtn = document.querySelector("#translate");  
120 const clearBtn = document.querySelector("#clear");  
121 const copyBtn = document.querySelector("#copy");  
122  
123 const select = document.querySelectorAll("select");  
124
```

- 複製與清除按鈕

```
136
137   clearBtn.addEventListener("click", () => {
138     input.value = "";
139     output.value = "";
140   });
141
142   copyBtn.addEventListener("click", () => {
143     navigator.clipboard
144       .writeText(output.value)
145       .then(() => {
146         console.log("已複製到剪貼板", output.value);
147       })
148       .catch(() => {
149         console.log("複製失敗");
150       });
151   });
152
```

- 這個 API 可以翻譯多國語言，做一個 select 選單，讓使用者選擇翻譯語言

demo > w18-api-project > JS app.js > ...

```
1  const countries = {  
2    "am-ET": "Amharic",  
3    "ar-SA": "Arabic",  
4    "be-BY": "Bielarus",  
5    "bem-ZM": "Bemba",  
6    "bi-VU": "Bislama",  
7    "bjs-BB": "Bajan",  
8    "bn-IN": "Bengali",  
9    "bo-CN": "Tibetan",  
10   "br-FR": "Breton",  
11   "bs-BA": "Bosnian",  
12   "ca-ES": "Catalan",  
13   "cop-EG": "Coptic",  
14   "cs-CZ": "Czech",  
15   "cy-GB": "Welsh",  
16   "da-DK": "Danish",  
17   "dz-BT": "Dzongkha",  
18   "de-DE": "German",  
19   "dv-MV": "Maldivian",  
20   "el-GR": "Greek",  
21   "en-GB": "English",  
22   "es-ES": "Spanish",  
23   "et-EE": "Estonian",  
24   "eu-ES": "Basque",  
25   "fa-IR": "Persian",  
26   "fi-FI": "Finnish",  
27   "fn-FNG": "Fanagalo",
```

```
28 "fo-FO": "Faroese",
29 "fr-FR": "French",
30 "gl-ES": "Galician",
31 "gu-IN": "Gujarati",
32 "ha-NE": "Hausa",
33 "he-IL": "Hebrew",
34 "hi-IN": "Hindi",
35 "hr-HR": "Croatian",
36 "hu-HU": "Hungarian",
37 "id-ID": "Indonesian"
```

API Project

翻譯

輸入要翻譯的文字

翻譯結果

Auto

 →

Amharic

翻譯

```
7 <meta name="viewport" content="width=device-width, initial-scale=1.0" />
8 <title>Translate</title>
9 <link rel="stylesheet" href="style.css" />
10 </head>
11
12 <body>
13 <h1>API Project <br><i class="fa-solid fa-language"></i> 翻譯</h1>
14 <div class="container">
15   <div class="input">
16     <textarea name="input" id="input" cols="30" rows="10" placeholder="輸入要翻
17   </div>
18   <div class="output">
19     <textarea readonly name="output" id="output" cols="30" rows="10" placehold
20   </div>
21   <div class="buttons">
22     <div>
23       <select id="from" class="dropdown">
24         <option value="auto">Auto</option>
25       </select>
26       <i class="fa-solid fa-arrow-right"></i>
27       <select id="to" class="dropdown">
28       </select>
29       <button id="translate">翻譯</i></button>
30     </div>
31     <div>
32       <button id="copy"><i class="fa-regular fa-copy"></i></button>
33       <button id="clear"><i class="fa-solid fa-trash"></i></button>
34     </div>
35   </div>
36   <script src="https://kit.fontawesome.com/9ce73c1069.js" crossorigin="anonymo
37   <script type="module" src="./app.js"></script>
38 </body>
39
40 </html>
```

- 用 JS 生成翻譯選項

```

123
124   const select = document.querySelector("select");
125
126   document.addEventListener("DOMContentLoaded", () => {
127       select.forEach((item) => {
128           for (const key in countries) {
129               const option = document.createElement("option");
130               option.value = countries[key];
131               option.textContent = countries[key];
132               item.appendChild(option);
133               // console.log(countries[key]);
134           }
135       });
136   });
137

```

實現翻譯功能

- 在 `translate()` 中呼叫 `optionEdit()` 設定 POST 的內容，並顯示結果。
- 因語言選項顯示的是語言，而 POST 中要使用語言代碼(ex:zh-tw)，所以在 `optionEdit()` 中先用 `for` 找出語言代碼

```

156 translateBtn.addEventListener("click", () => {
157     translate(input.value);
158 });
159
160 const optionEdit = (from, to, text) => {
161     for (const key in countries) {
162         if (countries[key] === from) from = key;
163         if (countries[key] === to) to = key;
164     }
165
166     options.body = JSON.stringify({
167         from: `${from}`,
168         to: `${to}`,
169         q: `${text}`,
170     });
171
172     console.log(options.body);
173 };
174
175 const translate = async (text) => {
176     if (text.length > 0) {
177         optionEdit(selectFrom.value, selectTo.value, text);
178         try {
179             const response = await fetch(url, options);
180             const result = await response.json();
181             console.log(result);
182             output.value = result;
183         } catch (error) {
184             console.error(error);
185         }
186     } else console.log("請輸入文字");
187 };

```

- 翻譯功能算是完成了，但語言選項為"auto"時，除了回傳翻譯結果還會回傳偵測到的語言種類，因此要另外處理

API Project

 翻譯

my name is Jack

我的名字是傑克,en

Auto



Chinese (Traditional)

翻譯



API Project

 翻譯

my name is Jack

我的名字是傑克

English



Chinese (Traditional)

翻譯



- 在 `translate()` 加入 `if` 判斷完回傳的數據再顯示

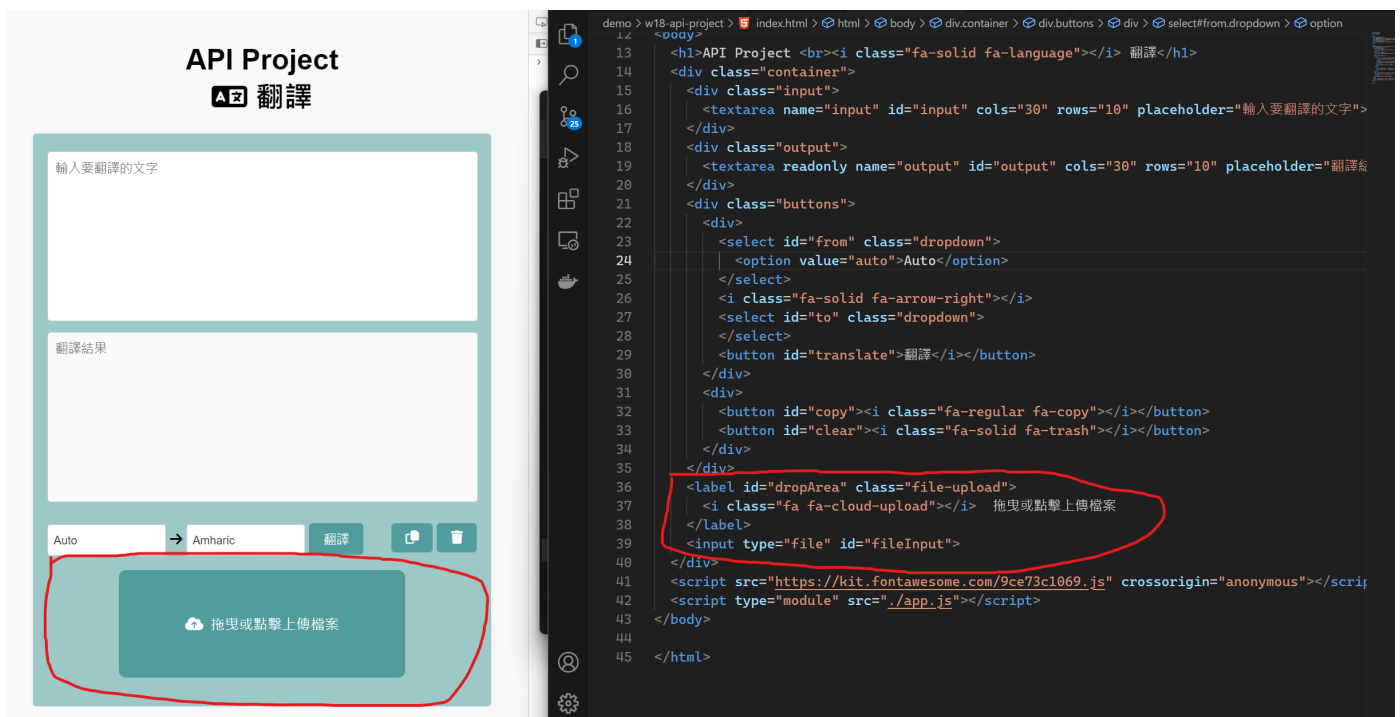
```

176
177 const translate = async (text) => {
178   if (text.length > 0) {
179     optionEdit(selectFrom.value, selectTo.value, text);
180     try {
181       const response = await fetch(url, options);
182       const result = await response.json();
183       console.log(typeof result[0]);
184       if (typeof result[0] === "string") {
185         output.value = result;
186       } else output.value = result[0][0];
187     } catch (error) {
188       console.error(error);
189     }
190   } else console.log("請輸入文字");
191 };
192

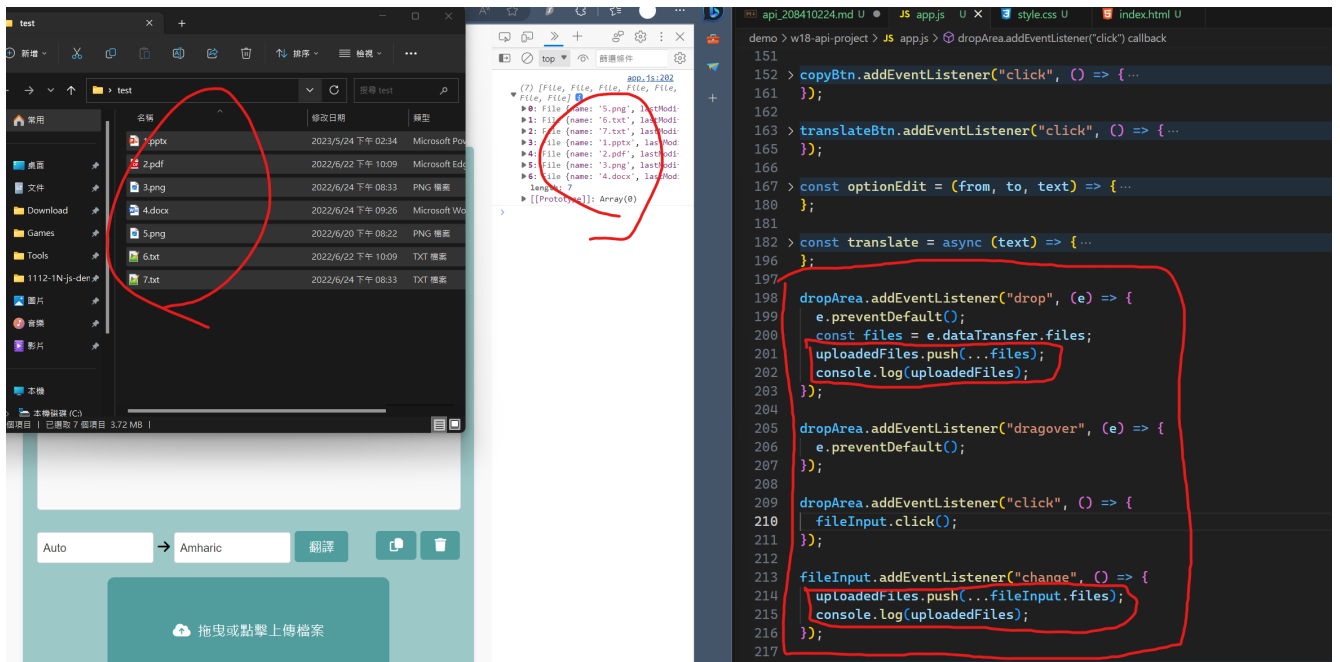
```

增加翻譯文檔的功能

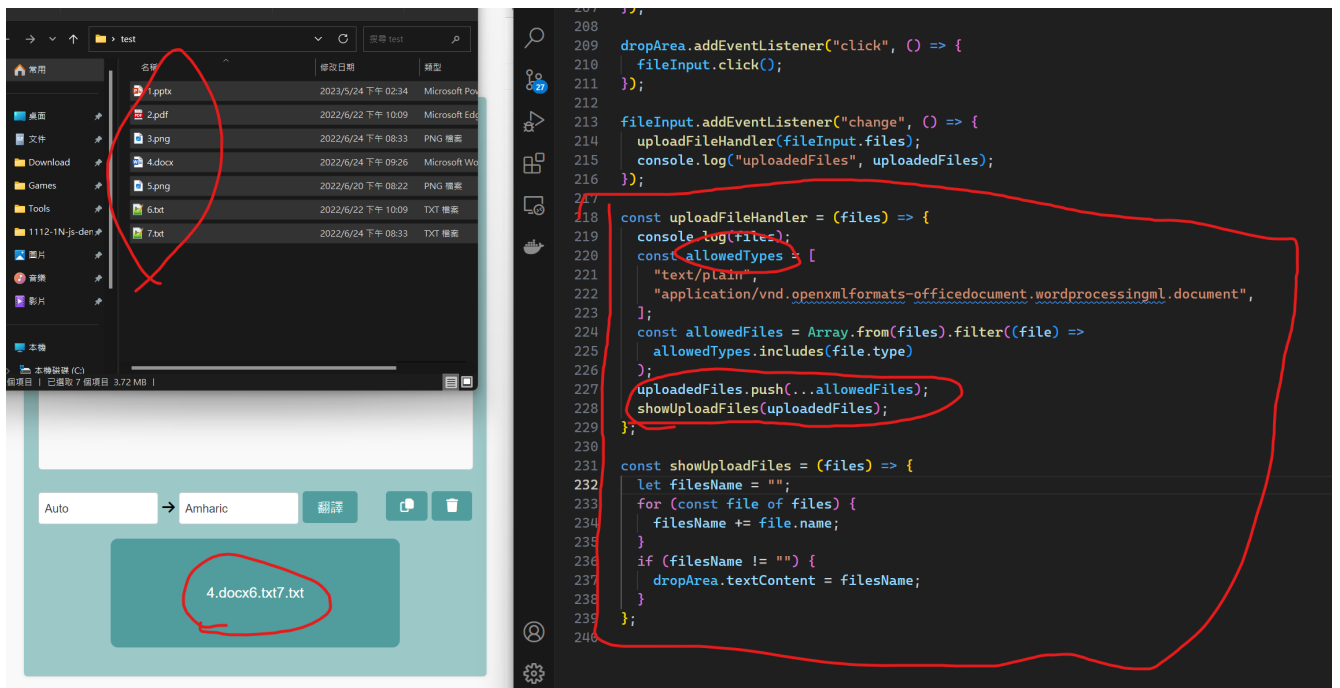
- 做出 UI 和 css



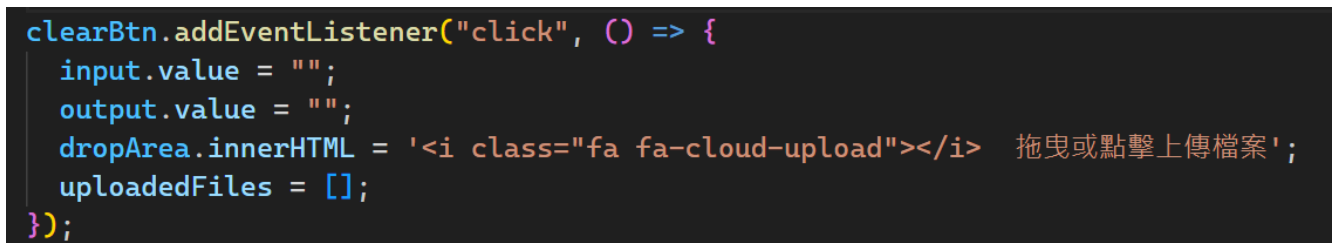
- 加入拖曳和選擇文件的功能
 - 新建一個 global variable uploadedFiles 存放拖曳或點擊上傳的文件
 - 用 `e.preventDefault()` 防止拖曳後跳窗



- 限制文件的種類(txt,docx)·並將上傳的文件顯示出來
 - 在 `uploadFileHandler()` 中用 `allowedTypes` 來篩選·將篩選後的文件放到 `uploadedFiles`·然後顯示



- 在 `clearBtn` 加入清除文件的功能



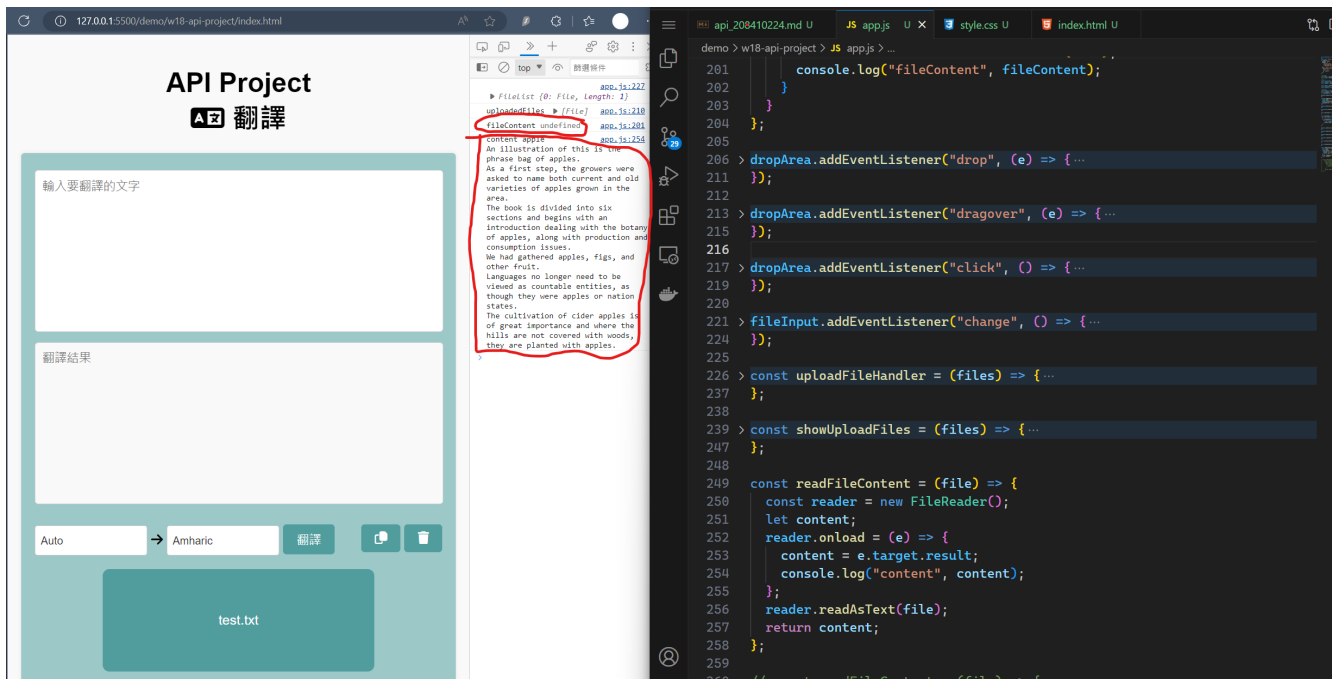
- 讀取文件內容
 - 按下翻譯後在 `translate()` 呼叫 `readFileContent()` 讀取上傳的文件


```

3
4 const translate = async (text) => {
5   if (text.length > 0) {
6     optionEdit(selectFrom.value, selectTo.value, text);
7   >   try { ...
4     } catch (error) {
5       console.error(error);
6     }
7   }
8   if (uploadedFiles.length > 0) {
9     for (const file of uploadedFiles) {
10      const fileContent = readFileContent(file);
11      console.log("fileContent", fileContent);
12    }
13  }
14 };
15

```

- 在 `readFileContent()` 中有成功 `console` 出文件內容，但 `reader.onload` 是非同步執行的，所以 `translate()` 中沒辦法接收在 `readFileContent()` return 的文件內容



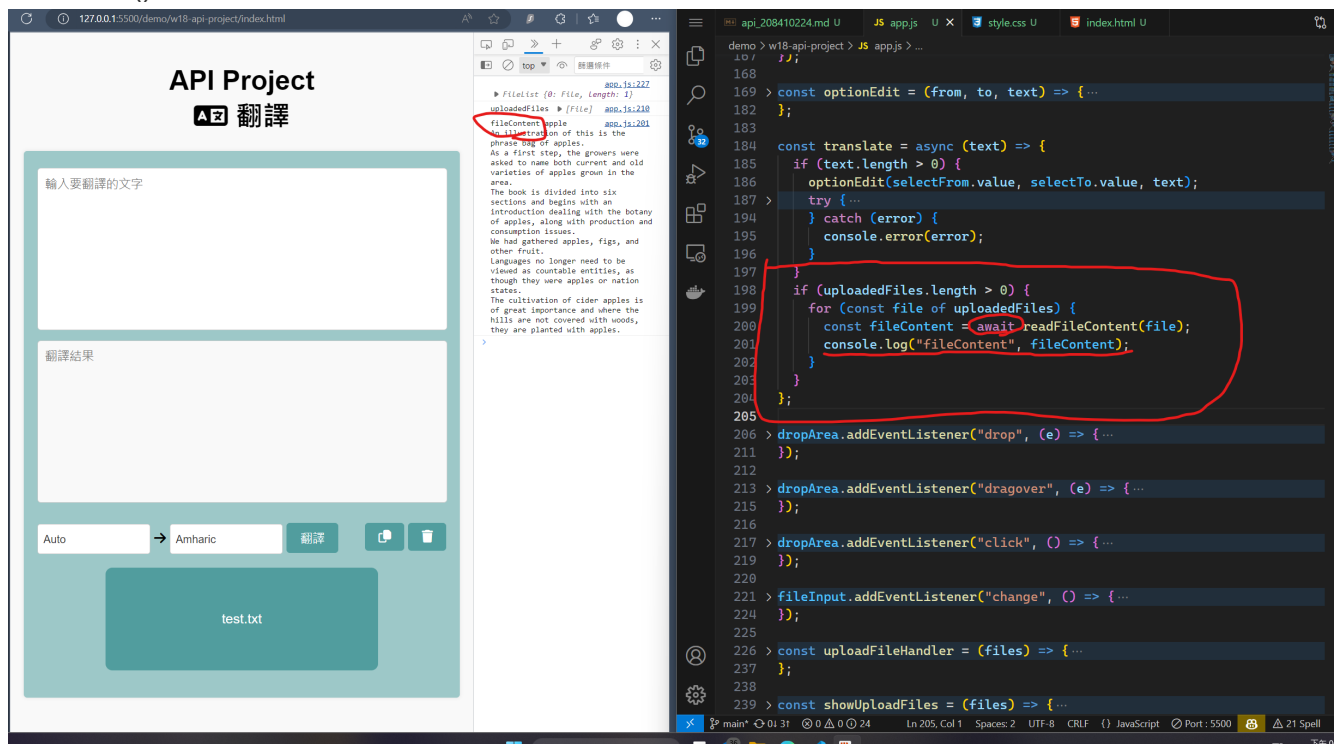
- 使用 `promise` 改寫 `readFileContent()`

```

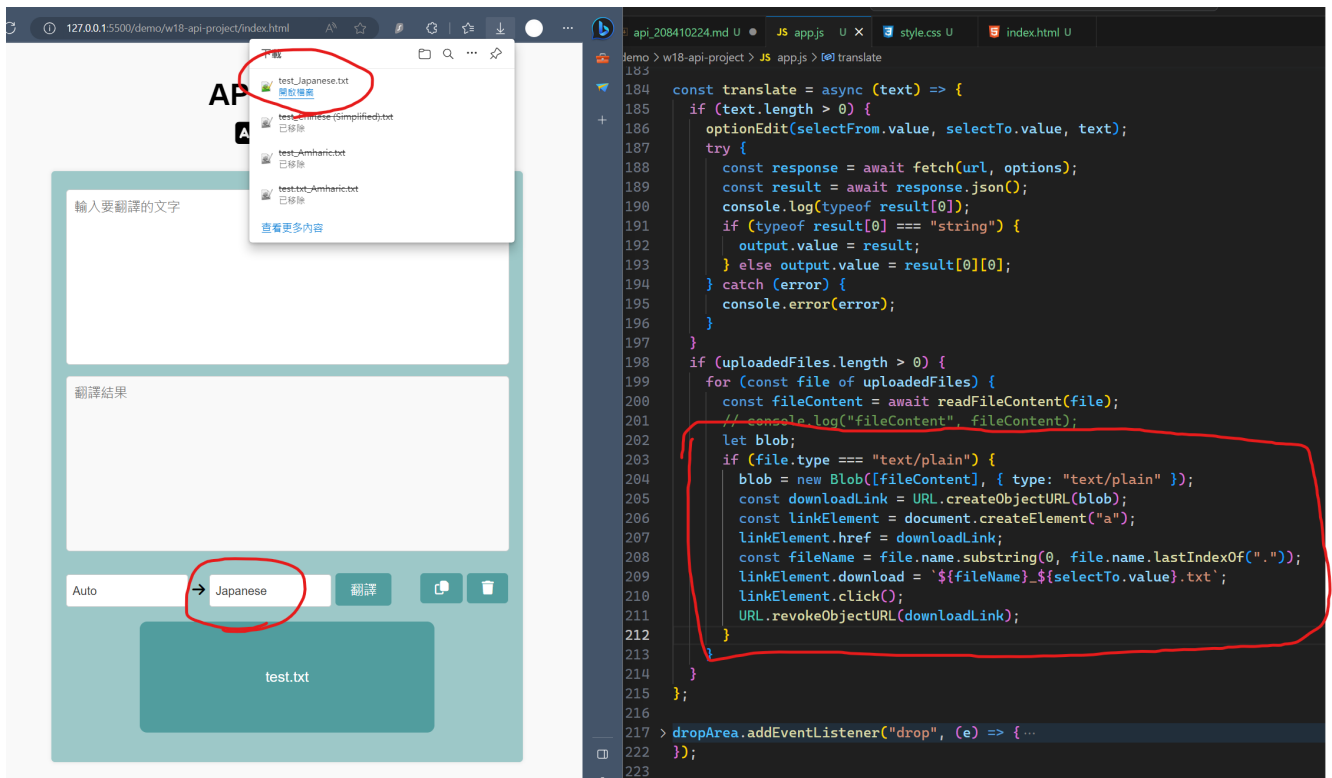
19  const readFileContent = (file) => {
20      return new Promise((resolve) => {
21          const reader = new FileReader();
22          reader.onload = (e) => {
23              const content = e.target.result;
24              if (content) {
25                  resolve(content);
26              }
27          };
28          reader.readAsText(file);
29      });
30  };
31

```

- translate()成功取得文件內容



- 下載文件
 - 下載功能實作，用 blob 存放翻譯後的內容，建立連結下載並將文件名加上翻譯的語言



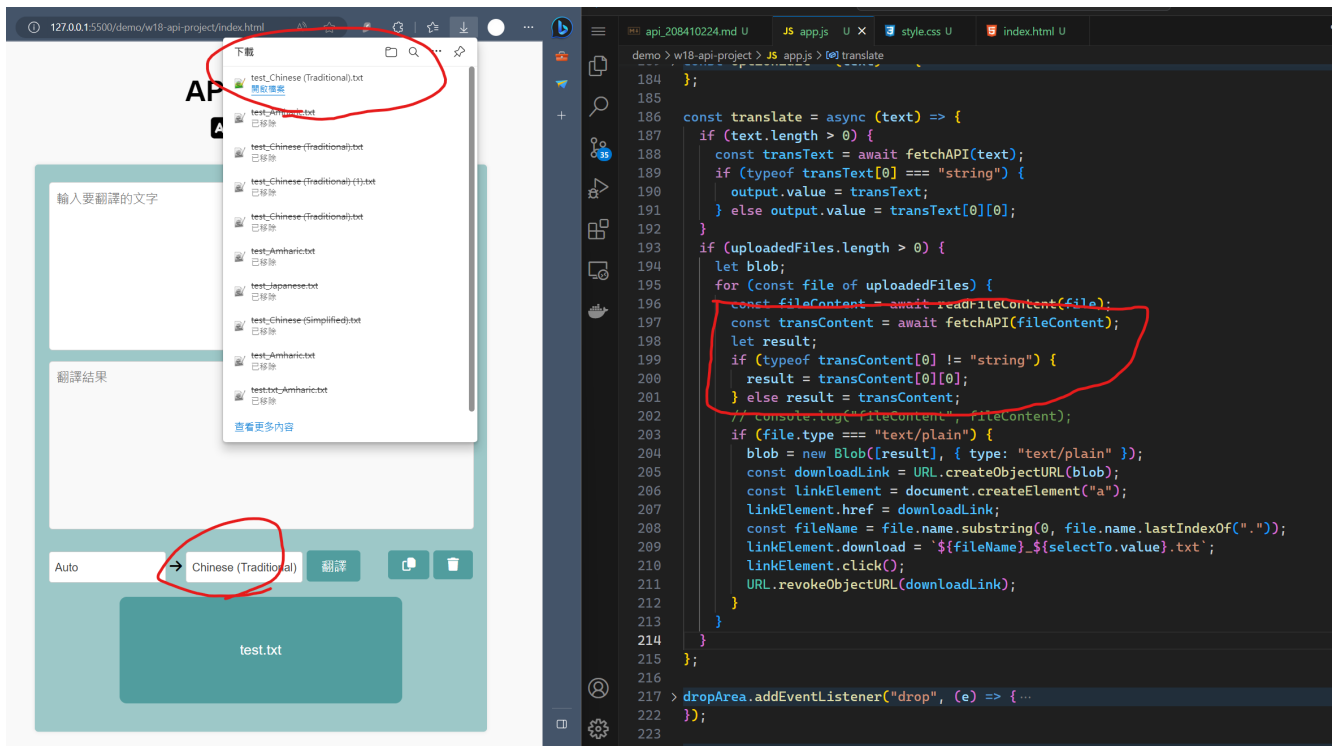
- javascript 沒有內建讀取 docx 的方法，需要使用 npm，放棄
- 完成翻譯文檔的功能
 - 因需要重複使用新增 fetchAPI()

```

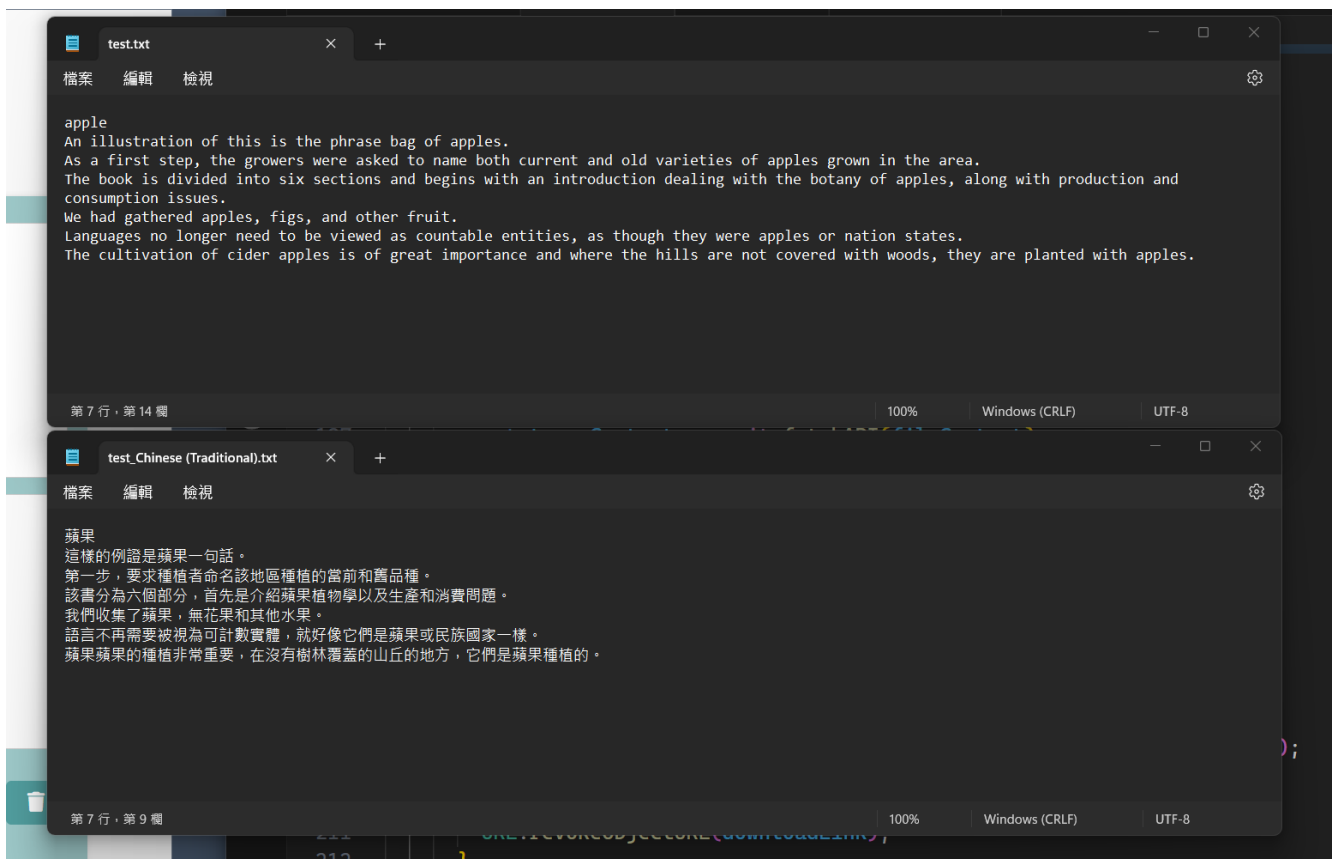
270  const fetchAPI = async (text) => {
271    optionEdit(text);
272    try {
273      const response = await fetch(url, options);
274      const result = await response.json();
275      return result;
276    } catch (error) {
277      console.error(error);
278    }
279  };
280

```

- 將翻譯的結果與下載功能結合



- 。上面是原檔，下面是翻譯結果



- 美化文件上傳後的 UI

API Project

 翻譯

輸入要翻譯的文字

翻譯結果

Auto



Amharic

翻譯



test - 複製 - 複製 (3).txt

test - 複製 - 複製.txt

test - 複製 (2).txt

test - 複製 (3).txt

test - 複製 - 複製 (2).txt

心得甘苦談

通過這項作業我更熟悉 api 的使用方法，整個實作過程雖然有碰到一些問題但也學習到了很多，Blob、上傳、下載是都是我沒有學過的方法，花了點時間去爬文，收穫很大。